PATTERN-THEORETIC AUTOMATIC TARGET RECOGNITION FOR INFRARED AND LASER RADAR DATA

A Dissertation
Presented to
The Academic Faculty

by

Jason Herbert Dixon

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
December 2015

PATTERN-THEORETIC AUTOMATIC TARGET RECOGNITION FOR INFRARED AND LASER RADAR DATA

Approved by:

Dr. Aaron Lanterman, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Patricio Vela
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Anthony Yezzi
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Brani Vidakovic
Wallace H. Coulter Department of Biomedical Engineering
Georgia Institute of Technology

Dr. Ghassan AlRegib
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: October 7, 2015

Dedicated to my family.

Without your constant love, support, and motivation,

I would not have made it this far.

ACKNOWLEDGEMENTS

First, I would like to acknowledge my research advisor, Dr. Aaron Lanterman. He has been a tremendously supportive mentor and guide through the doctoral process, especially in recent years as I finished the dissertation while working full-time away from campus. I thank Dr. Anthony Yezzi, Dr. Ghassan AlRegib, Dr. Patricio Vela, and Dr. Brani Vidakovic for serving on my dissertation committee. I also thank Dr. Russell Mersereau, Dr. John Cortese, and Dr. Erik Verriest for providing helpful comments and suggestions for my research after I completed my dissertation proposal.

I thank Dr. Gregory Arnold at the Air Force Research Laboratory (AFRL) for supporting this work. I also thank Dr. Kyle Erickson and Mr. Jeremy Olson for reviewing initial versions of the software I developed for some of the work in this dissertation and offering helpful comments and suggestions.

I must acknowledge the University of Maryland Baltimore County and the Meyerhoff Scholarship Program for providing me a wonderful undergraduate education and experiences that prepared me for graduate school.

I thank my wife, Dr. Rhea Brooking-Dixon, my parents, Herbert and Phoebe Dixon, and my sister, Dr. Stacey Dixon. They had been with me during the entire process, always believed in my potential, and acted as invaluable sources of encouragement, advice, tough-love, and sympathetic ears when I needed them. I thank my other friends and classmates from my days as a full-time student for making my experience at Georgia Tech much more enriching beyond academics and research. I also thank my co-workers for being flexible with my schedule as I finished my dissertation work.

Finally, I must acknowledge the various sources of funding that contributed to my

graduate education. These include graduate research and teaching assistantships from Georgia Tech's School of Electrical and Computer Engineering and the Georgia Tech President's Fellowship. Much of this work was sponsored by AFOSR Grant F49620-03-1-0340 entitled "Pattern-Theoretic Foundations of Automatic Target Recognition in Clutter."

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF SYMBOLS OR ABBREVIATIONS

3DS 3D Studio.

API Application Programming Interface.

ARL US Army Research Laboratory.

ATR Automatic Target Recognition.

CAD Computer-Aided Design.

CCD Charge-Coupled Device.

CNR Carrier-to-Noise Ratio.

CRLB Cramér-Rao Lower Bound.

DE Differential Evolution.

EM Expectation-Maximization.

FIM Fisher Information Matrix.

FLIR Forward-Looking Infrared.

FOV Field of View.

GA Genetic Algorithm.

GUI Graphical User Interface.

ICA Independent Component Analysis.

IF Intermediate Frequency.

LADAR Laser Radar.

MAP Maximum a Posteriori.

MCMC Markov Chain Monte Carlo.

MDA Multiple Discriminant Analysis.

MMSE Minimum Mean-Squared Error.

MuSES Multi-Service Electro-optic Signature.

OFF Princeton Shape Benchmark.

PCA Principal Component Analysis.

PRISM Physically Reasonable Infrared Signature Model.

SDE Stochastic Differential Equation.

STL STereoLithography.

SUMMARY

The problem of detecting and classifying objects of interest in images has been extensively studied. Pattern theory, a mathematical framework for representing knowledge of complex patterns developed by applied mathematician Ulf Grenander, has been shown to have potential uses in automatic target recognition (ATR). Prior research performed in the mid-1990s at Washington University in St. Louis resulted in ATR algorithms based on concepts in pattern theory for forward-looking infrared (FLIR) and laser radar (LADAR) imagery, but additional work was needed to create algorithms that could be implemented in real ATR systems. This was due to performance barriers and a lack of calibration between target models and real data. This dissertation addresses some of these issues by exploring techniques that can be used to create practical pattern-theoretic ATR algorithms.

This dissertation starts by reviewing the previous pattern-theoretic ATR research described above and discussing new results involving the unification of two previously separate outcomes of that research: multi-target detection/recognition and thermal state estimation in FLIR imagery. We then discuss ways of improving the overall utility of pattern-theoretic ATR by re-examining the following areas: 1) generalized diffusion processes to update target pose estimates and 2) the calibration of thermal models with FLIR target data.

As the number of target recognition algorithms increases, so does the need for accurate ways to compare the performance of one algorithm against another. Without a measure for performance, users of target recognition systems have no way of benchmarking one algorithm with another or determining if the sensor employed in the target recognition system can meet the necessary performance criteria for the

system. It is also useful to know if there is a fundamental limit on the ability to estimate a target's parameters of interest given a sensor's operating conditions. The final section of this dissertation analyzes the fundamental accuracy limits of target pose estimation under different sensor conditions, independent of the target detection/recognition algorithm employed. The Cramér-Rao lower bound (CRLB) is used to determine these accuracy limits.

CHAPTER I

INTRODUCTION

1.1 Automatic Target Recognition

Automatic Target Recognition (ATR) has been a topic of research interest for many years, employing many techniques that span the different areas of pattern recognition, computer vision, and statistical estimation and detection theory. The objective of ATR research is to create algorithms that can automatically detect and classify targets of interest, thus eliminating the need for a human to perform the same tasks.

Some of the more popular approaches used in ATR algorithms have been neural, statistical, or model-based in nature [44], with some algorithms resulting from a combination of these approaches. Such algorithms include statistical techniques like principal component analysis (PCA), multiple discriminant analysis (MDA), and independent component analysis (ICA) to differentiate between classes of targets or to synthesize targets from lower-dimensional representations. Various configurations of artificial neural networks or support vector machines [5] have also been used for target classification tasks. In one study, principal component analysis was combined with a multilayer perceptron to create a clutter rejector [12]. Some algorithms use databases of representative target templates and define metrics to match them against target data extracted from acquired imagery. Many of these techniques suffer from the need for training data to determine detection thresholds or classification decision regions. Classification or detection quality can deteriorate as the sample data becomes sufficiently different from the training data, or when the sample data is acquired under conditions that are not present in the training data. Detailed descriptions of these techniques can be found in references like [15]. Techniques using wavelets [54] and

mathematical morphology [4] also have been studied.

In the work presented in this dissertation, the ATR techniques are based on Ulf Grenander's pattern-theoretic framework, incorporating Bayesian statistics and Markov chain Monte Carlo (MCMC) techniques. The research efforts presented focus on automatically recognizing targets in imagery obtained from forward-looking infrared and laser radar sensors.

1.2 Pattern Theory

Ulf Grenander's work in pattern theory is the motivating force behind the ATR algorithms presented in this dissertation [25, 21, 24]. Pattern theory attempts to examine the underlying knowledge of patterns and their representations. Our goal is to quantitatively define those characteristics that are common to data received from a specific type of sensor and use that knowledge to match data against synthetic scenes created using deformable target models. This means that recognizing targets in a given scene is linked to synthesizing the scene containing those targets. While most computer vision and object recognition techniques focus on the separate stages of recognition (segmentation, feature extraction, classification, etc), the pattern-theoretic framework seeks to unify these separate concepts into a single process such that all steps are performed jointly. Another key feature of pattern theory is that the detection and recognition processes are performed directly on the data, thus acknowledging, from an information-theoretic standpoint, that no additional knowledge can be gained from preprocessing the data [10].

The basic approach to pattern-theoretic problem solving involves formulating the recognition task in a Bayesian framework. Targets of interest and their unique characteristics are parameterized and assigned a set of underlying probability distributions. The sensing apparatus may also contain a set of parameters and corresponding distributions. By formulating the problem in this way, the task of recognizing targets of

interest becomes a Bayesian estimation problem.

1.3 Sensor Models

The two sensor types considered in this study are forward-looking infrared (FLIR) and laser radar. Each has its own characteristics, which must be modeled when developing target recognition algorithms.

1.3.1 FLIR Sensors

The term FLIR represents passive imaging systems used to capture a representation of the thermal content in the sensor's field of view. FLIR sensors exploit the emissive characteristics of targets in the infrared portions of the electromagnetic spectrum. The amount of infrared radiation captured by the sensor is best thought of in the context of a difference between the target of interest and the background. FLIR sensors present unique challenges to ATR systems in that algorithm designers must account for the effects of temperature variability. A target of interest in FLIR imagery can appear in a number of different ways, depending on its internal operating state and condition of the environment. A target in motion may have a greater temperature than a target at rest and the gun of a tank that has recently been fired may have a greater temperature than one that has not been recently used. Time of day will affect sensor readings, as will significant changes in weather, like the presence of rain or fog. Depending on the severity of these thermal variations, targets can become practically indistinguishable from the background [31, 32]. Figure 1 shows a hypothetical synthetic FLIR scene.

Many ATR techniques have been developed specifically for FLIR data sets. For example, Chan et al. [6] of the U.S. Army Research Laboratory (ARL) have investigated ways of exploiting information from both the mid-wave and long-wave infrared bands in order to enhance target detection and clutter rejection performance. Their algorithms employed PCA and a supervised multilayer perceptron for classification

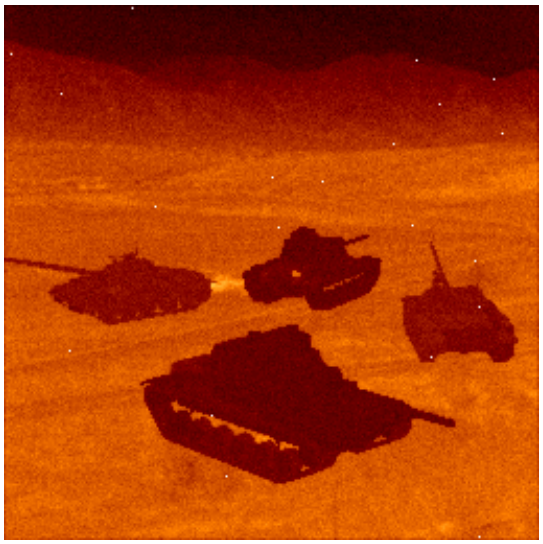


Figure 1: An example of a synthetic FLIR image corrupted by Gaussian noise and bad pixels.

tasks. Recognizing that individual ATR algorithms suffer various performance limitations, other work from ARL sought to combine different ATR algorithms to create a composite classifier, hoping that multiple algorithms working together could outperform any individual one [52].

Pattern-theoretic ATR methods account for sensor effects by using a likelihood function based on sensor statistics. Earlier pattern-theoretic FLIR ATR studies assumed a likelihood function based on Poisson statistics [40]. The FLIR sensor was taken to be a charge-coupled-device CCD producing Poisson distributed data with means proportional to the radiance of the objects in the sensor's field of view. Using such a model assumes the sensor is calibrated to give specific photon counts. While this is often true in astronomical imaging, it is usually not the case for operational FLIR sensors. Our current work uses a Gaussian model similar to the one discussed by Koksal et al. [36]. This model assumes the measured radiance is related to the true radiance of the objects present in the scene by Gaussian distributed noise consisting of a combination of thermal noise and shot noise. Measurements at each pixel are considered to be independent of all other pixels, so the loglikelihood function becomes

$$L_{IR}(\mathbf{d}|\boldsymbol{\mu}) = \sum_n \left\{ -\log\sqrt{2\pi\sigma_r^2} - \frac{(d(n)-\mu(n))^2}{2\sigma_r^2} \right\}, \quad (1)$$

where $\boldsymbol{\mu}$ is an ideal noiseless image, $\mathbf{d}$ is the measured data, and σ_r^2 is the variance of the measured radiance values [1]. We can interpret the Gaussian model with σ_r^2 as allowing us to accommodate various generic modeling mismatches beyond specific CCD sensor noise effects. The summation is computed over all pixels n in a given data image. In this study, we are only concerned with how the loglikelihood changes with different scene configurations, so we can ignore terms that do not depend on $\boldsymbol{\mu}$ and simply reduce the loglikelihood function per pixel to the squared error between the measured data and a hypothetical uncorrupted image $\boldsymbol{\mu}$.

1.3.2 LADAR Sensors

LADAR sensors are active imaging systems that can produce range imagery. LADAR data sets contain information about the range between the sensor and objects in the sensor's field of view. Many LADAR sensors work by emitting short pulses of light that reflect off the scene of interest and return to the sensor. The detectors in the sensor measure time of flight and use that information to determine range. Some laser radars also measure the intensity of the reflected light pulse. Figure 2 shows a hypothetical uncorrupted, synthetic LADAR scene.

Like FLIR ATR, there has been much work in the area of LADAR ATR. Many significant results have come from work done at the MIT Lincoln Laboratory. Koksal et al. have worked on statistical approaches for model-based LADAR ATR using maximum-likelihood and expectation-maximization techniques [37]. Spin-image

[1] We will assume that σ_r^2 is the same for every pixel, but it could readily be made a function of n without significantly changing any of the fundamental results of this dissertation. In particular, σ_r^2 could be made proportional to the data to form a weighted least-squares approximation of the Poisson likelihood function, as described in Section 3.3 of [41]

Figure 2: An example of a synthetic uncorrupted LADAR image.

based techniques have also gained popularity; Vasile et al. have used them for pose-independent ATR [62]. General pattern-theoretic ATR work is intended to be sensor-independent, except for the sensor-likelihood function. This dissertation focuses specifically on a sensor likelihood for LADAR.

To account for LADAR single-pixel noise statistics, we employed a coherent-detection likelihood function developed by Shapiro and Green [19, 20]. It has also been used in a multi-target ATR algorithm for LADAR data [39]. If $\mathbf{d}$ is an image of measured range values and $\boldsymbol{\mu}$ is an image of uncorrupted *true* range values, then the loglikelihood function is given by

$$L_{LR}(\mathbf{d}|\boldsymbol{\mu}) = \sum_n \log \left\{ [1 - Pr_A(n)] \frac{1}{\sqrt{2\pi\sigma^2(n)}} \exp \frac{-\{d(n) - \mu(n)\}^2}{2\sigma^2(n)} + \frac{Pr_A(n)}{R_{unc}} \right\}, \tag{2}$$

where $\sigma(n)$ is the local range accuracy for pixel n, given by

$$\sigma(n) = \frac{R_{res}}{\sqrt{\mathrm{CNR}(n)}}, \tag{3}$$

and $Pr_A(n)$ is the probability of anomalous measurement for pixel n, given by

$$Pr_A(n) = \frac{\log(N) - \frac{1}{N} + 0.557}{\mathrm{CNR}(n)}. \tag{4}$$

R_{unc} is the range uncertainty interval and N is the number of range resolution bins, given by

$$N = \frac{R_{unc}}{R_{res}}. \tag{5}$$

R_{res} is the range resolution, and CNR is the carrier-to-noise ratio, taken to be

$$\text{CNR}(n) = \frac{\eta P_T \rho A_R}{h\nu B\pi} \epsilon_{opt} \epsilon_{het} \frac{e^{-2\alpha\mu(n)}}{\mu^2(n)}, \tag{6}$$

where α is the atmospheric extinction coefficient, ϵ_{opt} is the receiver's optical efficiency, ϵ_{het} is the receiver's heterodyne efficiency, η is the detector's quantum efficiency, $h\nu$ is the photon energy, ρ is the target reflectivity, B is the intermediate frequency (IF) filter bandwidth, P_T is the peak transmitting power, and A_R is the receiver's aperture area. We will only consider the case with no anomalous measurements (i.e., $Pr_A = 0$), so the last term drops out of the loglikelihood function. It can now be reduced to

$$L_{LR}(\mathbf{d}|\boldsymbol{\mu}) = \sum_n -\frac{1}{2}\log\left\{2\pi\sigma^2(n)\right\} - \frac{[d(n)-\mu(n)]^2}{2\sigma^2(n)}. \tag{7}$$

The main ideas in this dissertation could be applied to the original Poisson FLIR likelihood model, or other quite different likelihood models, at the cost of the analytic and computational convenience afforded by Gaussian models.

1.4 Dissertation Organization

This section outlines the organization of this dissertation and the contents of each chapter.

1.4.1 Chapter 2 - Pattern-Theoretic ATR

Chapter 2 provides a brief history of automatic target recognition, discusses Ulf Grenander's work in pattern theory in the context of the ATR problem, and provides some examples of how previous Grenander-inspired jump-diffusion-based ATR algorithms performed before application of the improvements discussed in later chapters.

1.4.2 Chapter 3 - Non-Gradient Based Target Diffusion

Chapter 3 discusses modifying the diffusion process described in Section 2.1.4 to make the implementation adapt to viewing perspective and target characteristics. In previous implementations [38, 39], the stochastic differential equation that governs the refinement of initial pose estimates required two step sizes to be chosen. In the jump-diffusion experiments of [38, 39], both of these were determined empirically through a trial-and-error approach that yielded the best adjustments of the configuration parameters. In practice, these should be automatically determined. Chapter 3 considers an alternative algorithm. We discuss the mapping from real-world coordinates to two-dimensional image coordinates and how this mapping can be used to define a new type of diffusion that adjusts target position by individual pixel values. This mapping facilitates a search among the pose parameters within integer pixel units from the current position estimate, via Metropolis-Hastings style sampling. Using this scheme, adjustments can be made to the configuration parameters so that the changes in the estimated target pose parameters correspond to individual pixel sizes. We show that this scheme performs well for refining a target's estimated pose parameters.

1.4.3 Chapter 4 - Thermal State Inference with Infrared Calibration

The ATR algorithms for infrared imagery presented in Chapter 2 were tested on synthetic FLIR image sets. These image sets were created either directly from known thermal intensity values or indirectly by generating thermal intensity values probabilistically from a set of known thermal intensity values. This work assumed that the mapping between thermal intensities and pixel values provided by the detector match, or at least have a one-to-one deterministic mapping between them. In practice, this mapping is rarely known exactly. Even if carefully calibrated data of known targets can be taken before an operational situation, sensor characteristics can change over

time. Thus, it would be helpful to build the calibration problem into our pattern-theoretic formulation. This part of the dissertation research demonstrates a scheme to map thermal models to values that may be present in real FLIR imagery. Starting with the model for the thermal state estimation technique to be presented in Section 2.2.1, we adjust the model by assuming the relationship between the model and image produced at the detector is affine. This creates a nonlinear relationship among the desired parameters, and a new solution must be derived. Chapter 4 compares a traditional approach to solving a system of nonlinear equations to a novel approach that exploits the special structure in the equations produced by the affine model.

1.4.4 Chapter 5 - Accuracy Bounds on Pose Parameter Estimation

One characteristic that separates pattern-theoretic ATR from many other techniques is that pose estimation is a significant part of the target recognition process. Therefore, it is of interest to know how well target pose can be estimated given a sensor's operating conditions. This part of the dissertation employs Cramér-Rao lower bound analysis to determine fundamental accuracy limits on pose parameter estimation, assuming knowledge of the sensor likelihood function. We envision that an ATR system designer should be able to input sensor parameters and target models to determine bounds on the standard deviation of parameter estimators. We compare the bounds to the results of pose parameter estimation experiments using Monte Carlo trials and find many similarities between the theoretical bound and the simulations.

1.4.5 Chapter 6 - Conclusion

Chapter 6 summarizes the contributions resulting from the work presented in this dissertation. It also discuss topics that may be of interest in future pattern-theoretic ATR studies.

CHAPTER II

PATTERN-THEORETIC ATR

2.1 Pattern-Theoretic ATR

2.1.1 Representation of Complex Scenes

In following the pattern-theoretic philosophy, we must first define the elements within a scene. The patterns that we are interested in are built from templates. These templates may undergo transformations such as translations, rotations, changes of scale, or any other action that can be represented mathematically.[1] We will refer to the objects of interest within the imagery as *generators*. A *configuration* will denote the set of generators that make up the scene.

The set of generators in a scene configuration consists of an unknown number of vehicles. Each generator will contain knowledge about the object it represents, which in this case includes position, orientation, type, and, in the case of FLIR sensors, thermal state. Therefore, a single generator g representing a ground-based target in the set of generators $\mathcal{G}$ will be part of the configuration space $\mathcal{C}_1 = \Re^2 \times [0, 2\pi) \times \mathcal{A} \times \boldsymbol{\alpha}$, which defines a ground-based position and orientation, a generator class representing the type of target (e.g., $\mathcal{A}$={M2,M60,T62,...}), and a set of thermal parameters $\boldsymbol{\alpha}$ for FLIR imagery. A scene with N targets lives in a space $\mathcal{C}_N$. Since the number of targets is not known in advance, the full parameter space is the union $\mathcal{C} = \bigcup_{N=0}^{\infty} \mathcal{C}_N$.

2.1.2 Rendering Synthetic Imagery

This section presents our techniques for synthetic scene creation, which is a step required by pattern-theoretic ATR algorithms and for computing accuracy bounds

[1] In our formulation, the apparent changes of scale seen by the sensor are automatically handled via perspective projection, so we will not include an explicit scale parameter.

as discussed in Section 5. Objects are rendered from faceted models, composed of polygons (typically triangles) that make up components of the object's surface, using the OpenGL three-dimensional graphics application programming interface (API) [56, 1]. OpenGL performs the transformation $(\tilde{x}, \tilde{y}, \tilde{z}) \rightarrow (\tilde{x}/\tilde{z}, \tilde{y}/\tilde{z})$, taking a three-dimensional hypothesized scene and turning it into a two-dimensional image through perspective projection and obscuration. This transformation is accomplished by defining a set of matrices that take coordinates in three-dimensional object space and convert them to a corresponding set of two-dimensional image-space pixel positions.

The OpenGL coordinate transformation system is shown in Figure 3. Faceted models and scene components consist of a set of object coordinates, $[x, y, z]$ tuples that represent the boundaries for the different facets that make up an object's surface. Transforming the object coordinates via a model-view matrix results in the object coordinates being transformed into eye coordinates. Eye coordinates represent an object with respect to some common measurement system. The eye coordinates are then transformed by a projection matrix into clip coordinates. Since we use perspective projection, the clip coordinates represent positions within a frustrum that represents the part of three-dimensional space in our final image. The clip coordinates are transformed into normalized device coordinates by dividing by the clipping range so that the all scene coordinates are in the range $[-1, 1]$. The normalized device coordinates can finally be mapped to window coordinates through a viewport transformation. The window coordinates represent the location where that part of the three-dimensional object appears on the two-dimensional window.

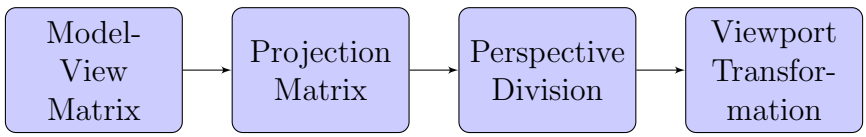


Figure 3: Coordinate transformation process in OpenGL.

Target configurations used in this dissertation consist of a combination of M2,

M60, and T62 faceted tank models derived from the PRISM software package.[2] We assume a tank rests on a flat ground plane at some position (x, y)[3] and orientation angle θ, and has a type $a \in \mathcal{A} = \{M2, M60, T62\}$.

The creation of synthetic FLIR and LADAR imagery starts with positioning the target models in a desired configuration, followed by making appropriate OpenGL calls to render the three-dimensional scene. More details on the scene rendering process can be found in Section A.2. For FLIR imagery, each facet of the target's surface is set to an intensity value corresponding to the target's thermal profile. The targets are then placed over an infrared background image. The background image was included to approximate what a real infrared scene may look like, but no effort was made to relate the viewing parameters of the background to the viewing parameters of the rendered images.

We simulate ideal LADAR images by exploiting the depth buffering system employed by OpenGL. The first step is to read the values that OpenGL stores in its depth buffer. Next, these values and the (x', y') two-dimensional pixel coordinates for the corresponding points in the image can be used to undo the perspective projection and obtain the original $(\tilde{x}, \tilde{y}, \tilde{z})$ vectors of three-dimensional world coordinates for the scene.[4] The uncorrupted range values are determined by computing the distance from the viewing location to those particular coordinates, producing a range image.

Test data sets are created by corrupting the rendered images with noise as defined by the sensor's likelihood function.

---

[2]The PRISM software, originally developed by the Keweenaw Research Center at Michigan Technological University, was sold and maintained by ThermoAnalytics, Inc., P.O. Box 66, Calumet, MI 49913, web: www.thermoanalytics.com. It has been replaced by the MuSES infrared signature prediction software.

[3]We use tildes earlier to indicate that the coordinate system is transformed according to the camera position and orientation.

[4]Tildes are used to indicate the three-dimensional world coordinates used when rendering scenes in OpenGL, whereas primes indicate image pixel coordinates.

2.1.3 Sensor Statistics and the Posterior Distribution

The techniques considered here for detecting and recognizing targets of interest follow a Bayesian framework. We start with a likelihood function that models sensor statistics and use it to compare scenes created from hypothesized configurations with the collected data. The likelihood is combined with prior information to form a Bayesian posterior distribution. We consider uniform priors over position and orientation. In practice, we implicitly introduce the prior information that two targets may not occupy the same space by not considering such configurations. The sensor likelihood functions for FLIR and LADAR, previously defined in Sections 1.3.1 and 1.3.2, are used with the posterior distribution.

Given an estimated configuration state c, the likelihood and the prior combine to form a Gibbs posterior distribution of the form

$$\pi(c|\mathbf{d}) \propto \exp[H(c|\mathbf{d})], \tag{8}$$

where $H(c|\mathbf{d}) = L(\mathbf{d}|c) + P(c)$ is the logposterior created by summing the loglikelihood $L(\mathbf{d}|c)$ and the logprior $P(c)$. $L(\mathbf{d}|c) = L_{SENSOR}[\mathbf{d}|\text{render}(c)]$, where render$(c)$ represents the process of obtaining an uncorrupted image $\boldsymbol{\mu}$ from a scene configuration c through perspective projection and obscuration. The distribution (8) represents how closely related the hypothesized configuration is to the data image. We sample it by a reversible jump Markov chain Monte Carlo procedure called a jump-diffusion process. The next section explores the jump diffusion process in more detail.

2.1.4 Inference by Metropolis-Hastings Jump-Diffusions

The full posterior distribution represented by (8) quantifies the relationship between the data and all possible hypothesized scenes, and is thus a function of several parameters (e.g., pose and target class) with a dimension that changes with the number of targets in the hypothesized scene. To estimate the parameters of interest, we need a way to sample from this posterior distribution that allows us to adjust the number

of targets, since we do not know this information in advance. We follow the jump-diffusion framework presented in [40], Miller et al. [48, 47], and Srivastava et al [58]. The algorithm is designed around a reversible jump Markov process that accounts for the continuous and discrete aspects of the ATR problem. The discrete components are handled by *jumping* from one subspace to another, deciding whether to add a target, remove a target, or change a target's type. These choices will henceforth be called *birth*, *death*, and *metamorph*, respectively. The continuous aspects, namely, the inference of the pose parameters, are refined via a diffusion process. More recently, jump-diffusion processes have been applied to range image segmentation [28], estimating medial axes of two-dimensional shapes [64], building layouts using radar [49], and tracking maneuvering targets with radar [60].

The decision process used when determining whether or not to accept a jump involves random sampling and Metropolis-Hastings acceptance/rejection [29]. For birth and metamorph jumps, a representative set of candidate locations and orientation angles is chosen, respectively. For death jumps, candidates are chosen by removing a single target from the hypothesized configuration. The posterior probabilities are computed for each of these candidates, and one candidate is randomly chosen with a probability proportional to its posterior probability. In practice, one candidate posterior probability typically dominates over the others, so it often *appears* as though the candidate with maximum probability is automatically chosen. The Metropolis-Hastings algorithm accepts the chosen candidate with probability

$$\beta(c_{orig}, c_{prop}) = \min\left(\frac{\pi(c_{prop})r(c_{prop}, c_{orig})}{\pi(c_{orig})r(c_{orig}, c_{prop})}, 1\right), \tag{9}$$

where c_{prop} is the proposed state of the configuration, while c_{orig} is the current state. The functions $r(c_{prop}, c_{orig})$ and $r(c_{orig}, c_{prop})$ are the transition probabilities; see [40] for details. The function $\pi(c)$ is the probability of being in state c, which in this case is derived from the logposterior $H(c|\mathbf{d})$.

The type of jump to perform is determined probabilistically by a prior distribution

based on the number of hypothesized targets in the configuration. As is typical with continuous-time Markov processes, the time between jumps is exponentially distributed. During the intervals between jumps, the diffusion process perturbs the continuous pose parameters by small amounts to better align the hypothesized targets with the data. Diffusions are accomplished using the Langevin stochastic differential equation (SDE):

$$d\mathbf{C}_N(\tau) = \nabla_{\mathbf{C}_N} H[\mathbf{C}_N(\tau)|\mathbf{d}]d\tau + \sqrt{2}dW_N \tag{10}$$

where W_N is a Wiener process and $H[\mathbf{C}_N(\tau)|\mathbf{d}]$ is the logposterior associated with the configuration parameter vector $\mathbf{C}_N$, which contains the configuration parameters for N targets of fixed classes. The time index τ refers to a unit of time within the diffusion interval. Once (10) is discretized, τ can simply be thought of as a discrete time index, so a finite number of diffusion steps occur between jumps, and that number is an exponential random variable.

For a more detailed analysis of the theory behind jump-diffusion processes in general, refer to the aforementioned works in [40], Miller et al. [47], and Srivastava et al [58].

2.1.5 Jump-Diffusion Implementation Details

The derivative needed in solving the Langevin SDE (10) may be computed with a finite difference approximation:

$$\frac{\partial H(c|\mathbf{d})}{\partial c_p} \approx \frac{H(\ldots, c_p + \delta, \ldots|\mathbf{d}) - H(\ldots, c_p - \delta, \ldots|\mathbf{d})}{2\delta} \tag{11}$$

where c_p is an element of the configuration c, δ is some small deviation of the parameter c_p, and the ellipses indicate the remaining parameters are held fixed.

The Gibbs-form posterior distribution is explored by sampling the space of possible configurations with respect to the parameters of interest, which involves rendering hypothesized scenes during any birth, death, or metamorph move. In determining the acceptance probability for any move, we obtained good results by assuming that

the forward and reverse transition probabilities are equal and by simply comparing the proposed and original posterior probabilities of the hypothesized configuration scenes. For a more intuitive sense of the implementation of the jump-diffusion process, see the block diagram in Figure 4.

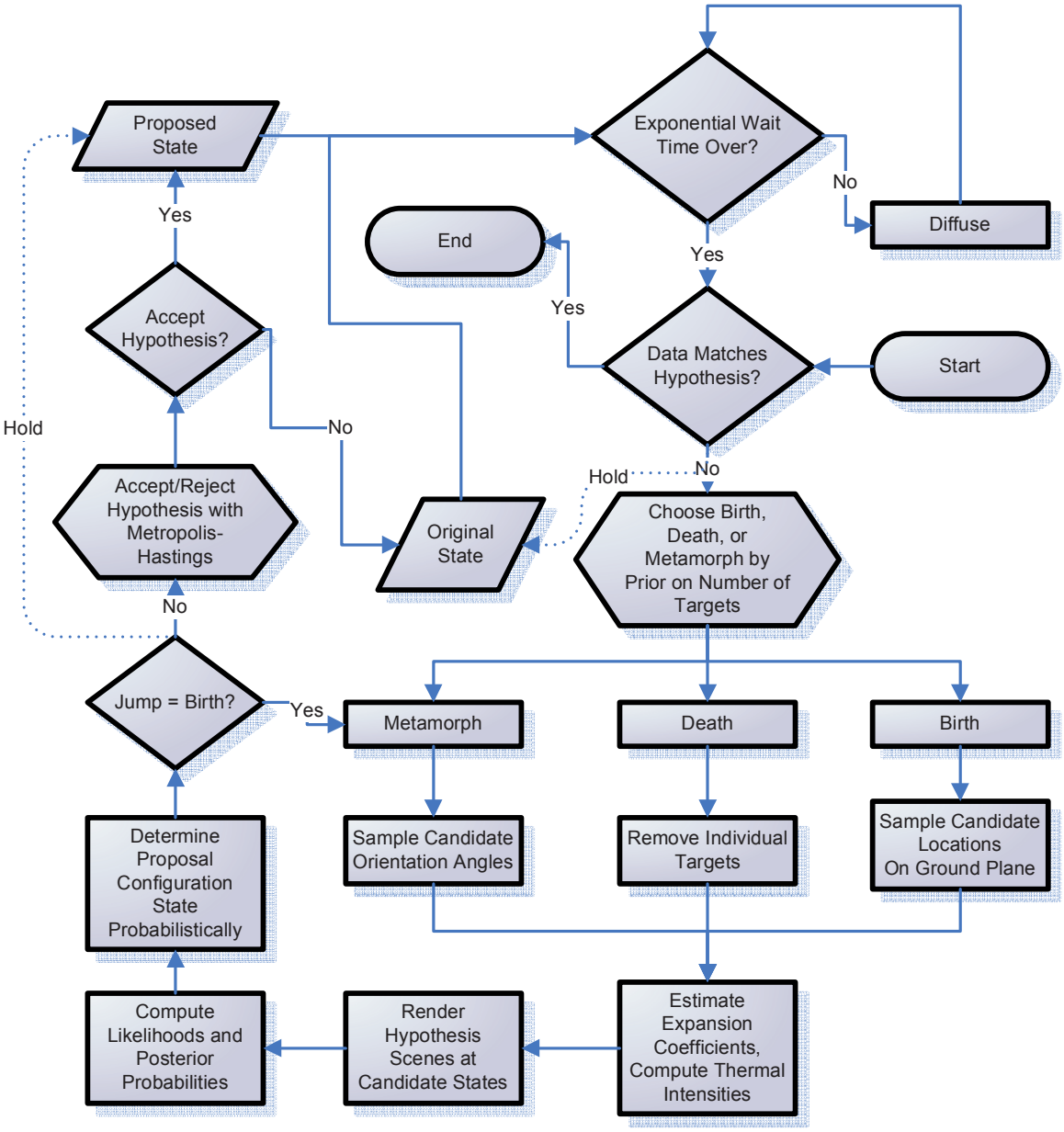


Figure 4: Block diagram for implementing the jump-diffusion process for ATR in infrared data.

2.2 Representing Thermal Variability

The jump-diffusion discussion presented in Section 2.1 focused on the general principles for the pattern-theoretic ATR algorithm without considering any specific sensor. When working with data imaged from FLIR sensors, thermal variability also must be considered. This section presents a technique used to estimate the thermal states of recognized targets at each iteration of the ATR algorithm.

2.2.1 Eigentanks

We model the thermal variability of targets by constructing prior distributions on the radiant intensities of target facets [8, 9, 41]. The PRISM software was used to simulate radiance by CAD models of ground-based targets. By simulating a large number of radiance measurements, taken while varying environmental and internal heating parameters over reasonable ranges, a set of typical radiance profiles was generated, to which principal component analysis was applied. The first few eigenvectors, here called *eigentanks*, from the principal component analysis are used to estimate the thermal profiles of targets in the data.[5]

The surface of the CAD model of the tank is divided into I regions, with the intensity assumed constant across each region, and J basis functions are employed. Let A_i denote the surface area of region i and λ_i represent the intensity of region i. Representations of the form $\lambda_i = \sum_j \alpha_j \Phi_{ij} + m_i$ are employed, where m_i is the mean of region i, and Φ_{ij} is eigentank j at region i. The α_j terms are the expansion coefficients. The eigentanks and associated eigenvalues γ_j were determined using a sample covariance matrix of the set of typical radiance profiles weighted by the surface area A_i of the corresponding regions. A complete derivation may be found in [41].

[5]The PRISM databases and resulting principle component models employed in the experiments discussed here were created by Matthew Cooper [8, 9].

2.2.2 Estimation of Expansion Coefficients

This discussion will follow a derivation found in [41], but will employ our Gaussian data likelihood instead of a Poisson likelihood. Consider a collected data set $\mathbf{d}$. Let N_i denote the number of pixels in region i, as seen by the sensor, and $D_i = \sum_{k \in R_i} d(k)$ be the sum of data pixels in region i. Conditioned on the α_j terms, we have the relationship $d(k) \sim Gaussian(\lambda_i, \sigma_r^2)$ for $k \in R_i$. In accordance with the principal component analysis mentioned in Section 2.2.1, a Gaussian prior is placed on the α_j terms, with variances given by the eigenvalues found from the analysis. Dropping terms independent of α_j, the logposterior for the pixels on target in terms of the expansion coefficients is

$$H(\alpha|D) = -\sum_i \sum_{k \in R_i} \frac{(\lambda_i - d(k))^2}{2\sigma_r^2} - \sum_j \frac{\alpha_j^2}{2\gamma_j} \tag{12}$$

$$= -\sum_i \sum_{k \in R_i} \frac{\lambda_i^2 - 2\lambda_i d(k) + d^2(k)}{2\sigma_r^2} - \sum_j \frac{\alpha_j^2}{2\gamma_j} \tag{13}$$

$$= -\sum_i \frac{N_i\lambda_i^2 - 2\lambda_i \sum_{k \in R_i} d(k) + \sum_{k \in R_i} d^2(k)}{2\sigma_r^2} - \sum_j \frac{\alpha_j^2}{2\gamma_j}. \tag{14}$$

Incorporating $\lambda_i = \sum_j \alpha_j \Phi_{ij} + m_i$ and taking derivatives with respect to each α_j, we obtain equations of the form

$$\frac{\partial}{\partial \alpha_j} H(\alpha|D) = -\sum_i \frac{2N_i\lambda_i \frac{\partial}{\partial \alpha_j}\lambda_i - 2\frac{\partial}{\partial \alpha_j}\lambda_i \sum_{k \in R_i} d(k)}{2\sigma_r^2} - \frac{\alpha_j}{\gamma_j} \tag{15}$$

$$= -\sum_i \frac{N_i(\sum_k \alpha_k \Phi_{ik} + m_i)\Phi_{ij} - \Phi_{ij} D_i}{\sigma_r^2} - \frac{\alpha_j}{\gamma_j}. \tag{16}$$

To maximize the logposterior, we must satisfy these J necessary conditions:

$$-\sum_i \frac{N_i(\sum_k \alpha_k \Phi_{ik} + m_i)\Phi_{ij} - \Phi_{ij} D_i}{\sigma_r^2} - \frac{\alpha_j}{\gamma_j} = 0, \ \forall j \tag{17}$$

$$-\sum_k \alpha_k \sum_i \frac{N_i \Phi_{ik} \Phi_{ij}}{\sigma_r^2} - \sum_i \frac{N_i \Phi_{ij} m_i - \Phi_{ij} D_i}{\sigma_r^2} - \frac{\alpha_j}{\gamma_j} = 0, \ \forall j \tag{18}$$

$$-\sum_k \alpha_k \sum_i \left[N_i \Phi_{ik} \Phi_{ij}\right] - \sum_i \Phi_{ij} \left[N_i m_i - D_i\right] - \frac{\sigma_r^2}{\gamma_j}\alpha_j = 0, \ \forall j. \tag{19}$$

Fortunately these are linear equations, conveniently expressed in matrix form:

$$\left(\begin{bmatrix} \sum_i N_i \Phi_{i1}^2 & \cdots & \sum_i N_i \Phi_{iJ} \Phi_{i1} \\ \vdots & & \vdots \\ \sum_i N_i \Phi_{i1} \Phi_{iJ} & \cdots & \sum_i N_i \Phi_{JJ}^2 \end{bmatrix} + diag \begin{pmatrix} \frac{\sigma_r^2}{\gamma_1} \\ \vdots \\ \frac{\sigma_r^2}{\gamma_J} \end{pmatrix} \right) \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_J \end{bmatrix} = \begin{bmatrix} \sum_i \Phi_{i1}(D_i - N_i m_i) \\ \vdots \\ \sum_i \Phi_{iJ}(D_i - N_i m_i) \end{bmatrix}. \quad (20)$$

For a given target pose, these equations allow us to compute approximate maximum a posteriori (MAP) estimates for the α_j terms in closed form. Note that α_j changes with different poses, as the MAP estimate adjusts to best match the data under the constraint of the eigentank model. For a multiple target scene, we perform a similar calculation for each hypothesized target.

To determine the N_i and D_i terms required for computing the expansion coefficients in (20), we use a *paint by numbers* technique [41]. Each target in a configuration is rendered with a set of increasing, yet disjoint, region numbers so that each intensity region is colored by a different number. One can easily compute N_i by counting the pixels of a common region number and D_i by summing the corresponding data pixels. After computing the expansion coefficients, the inferred intensities can be used to color in the image of region numbers. We consider the background to be of constant intensity equal to the average background intensity of the data. This final image is considered the hypothesized uncorrupted FLIR scene and is compared to the data set through the FLIR loglikelihood function.

2.2.3 Jump-Diffusion with Thermal Inference

This section describes our effort to combine the jump-diffusion algorithm from Section 2.1.4, originally discussed in [40], with the thermal state inference described in Section 2.2, originally discussed in [41], to show that the jump-diffusion algorithm

can be easily modified to accommodate the estimation of thermal states for detected targets. After a hypothesized target's position, orientation, and type have been chosen, (20) may be solved to determine the estimated thermal state. In terms of the block diagram in Figure 4, this means inserting a block for thermal state estimation after rendering the hypothesized configuration but before computing the likelihoods and posterior probabilities.

Results from a jump-diffusion simulation are shown in Figure 5. Figure 5(a) shows the FLIR data set used in this simulation, which consists of four tanks. The highest and lowest tanks in the image are both M60s and the remaining two are T62s. Each was initialized with a different position, orientation, and thermal state. The thermal states were synthesized using our Gaussian PCA models. We assume the camera viewing parameters are known. The algorithm begins by searching over the configuration space for the best set of parameters for a single new target. In the subsequent images, the white tanks represent the estimated configuration at that iteration. Figure 5(b) shows that the first tank located is the M60 that is positioned closest to the FLIR sensor. Since this tank has the greatest number of pixels on target, it makes sense that the algorithm would choose that position for a tank to achieve the greatest gain in likelihood.

The next few images shown in Figures 5(c)-5(e) show how the algorithm subsequently detects and places new hypothesized targets over the existing targets in the data set. Between the birth, death, and metamorph moves, the algorithm diffuses over the existing targets in an attempt to refine their pose parameters. The estimated thermal profiles of the hypothesized targets change as the diffusions take place because the adjustments to pose change the overlap with the corresponding target in the data image.

Figures 5(f) and 5(g) are interesting because they demonstrate the flexibility of the jump-diffusion process. A metamorph move occurs before the rightmost T62

fully diffuses over the T62 in the data set, and the best orientation angle turns out to be in the opposite direction (the estimated tank no longer points away from the FLIR sensor as shown in the data). Once the diffusion allows the estimated T62 to noticeably cover the T62 in the data image, another metamorph move brings it into the appropriate alignment, as shown in Figure 5(g). The final estimate is shown in Figure 5(h). All of the targets are aligned with the corresponding targets in the data image. The estimated thermal profiles also match those found in the data.

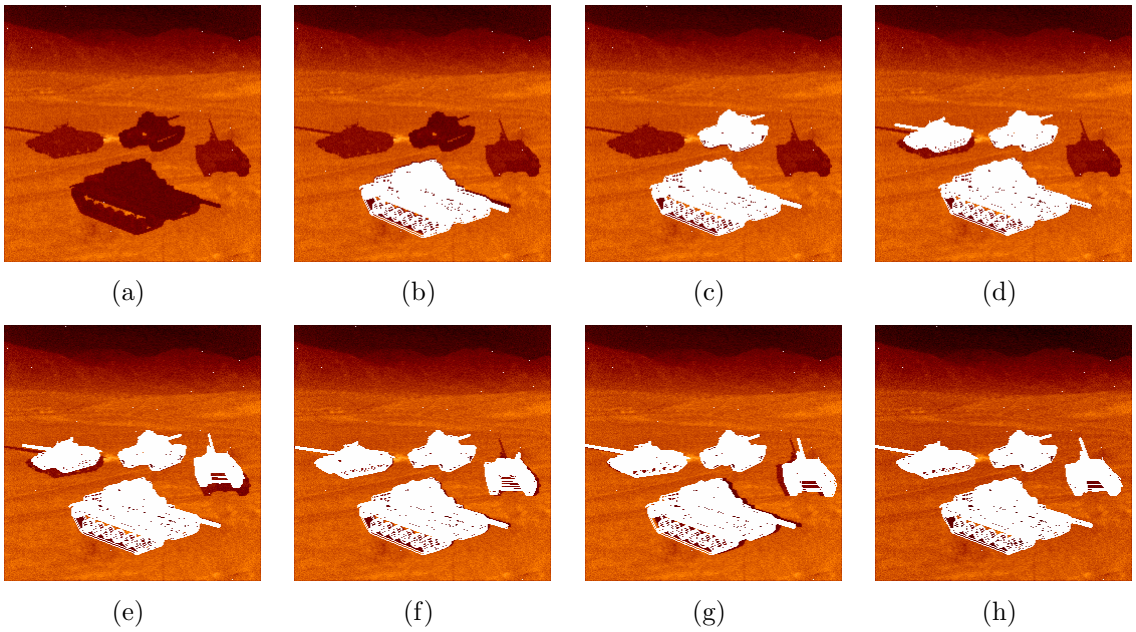


Figure 5: Intermediate results of a jump-diffusion process with thermal state inference for FLIR ATR.

There is some dissonance between the way we are approaching the pose parameters and the thermal state parameters. We compute the MAP estimate of the thermal state parameters, but draw random samples from the posterior distribution for the pose parameters, from which one might compute, for instance, an MMSE or MAP estimate. To be consistent, one could diffuse over the thermal state parameters as well as the pose parameters, although this would waste the advantage provided by relatively easy MAP estimation of the thermal state parameters. Plugging the MAP

estimates of the thermal parameters into the posterior distribution to yield a simplified posterior expression is analogous to generalized ML approaches in frequentist frameworks. A more theoretically satisfying approach would be to fully "integrate out" the thermal state parameters, perhaps using Laplace's method as in [39]. We leave the integration of such techniques into our new algorithms as an avenue for future work.

We have found that the posterior distributions tend to be so heavily peaked that diffusion-type algorithms for these sensors tend to strongly resemble a fairly direct march towards a MAP estimate, and they tend to cling closely to the peak once they are there, even though randomness is inherent in the algorithms.

2.3 Conclusion

This chapter provided an overview of pattern theory in the context of automatically recognizing targets of interest in infrared and laser radar imagery. We reviewed some of the current work in automatic target recognition, discriminating between our approach that those of others. We ended with some results showing how thermal state inference can be easily included into the existing pattern-theoretic jump-diffusion framework.

CHAPTER III

NON-GRADIENT BASED TARGET DIFFUSION

3.1 Introduction

This chapter discusses a modification to the diffusion process used in our pattern-theoretic ATR algorithm to facilitate robust pose parameter estimation under varying viewing perspectives. The stochastic differential equation (10) that governs the refinement of initial pose estimates requires choosing two step sizes: one step size for the derivative computation, and another step size for the discretized diffusion in each dimension. In previous jump-diffusion experiments [40, 39], both of these were determined empirically through a trial-and-error approach that yielded the best adjustments to the target pose parameters. While manually choosing an appropriate step size for a particular iterative algorithm is a common practice in gradient-based optimization and estimation problems [35, 38], it introduces additional complexity in pattern-theoretic ATR, because appropriate step sizes tend to depend on the target type and the viewing geometry of the scene containing the target.

For the two position parameters (x, y) specified in Section 2.1.2, effective step sizes tend to increase with both the viewing distance along the pose parameter axis and the distance between the target and the sensor. Difficulties appear when adjusting the step size while simultaneously moving the sensor closer to the target while widening the field of view. When considering the orientation angle θ, which does not have a viewing axis like the position parameters (x, y), the diffusion step size is most influenced by the resolvable two-dimensional target features as imaged by the sensor. These features depend on the distance between the sensor and target, the image

resolution, and the detail in the target model. In practice, implementation of pattern-theoretic techniques requiring refinement of continuous parameters via a diffusion process may become prohibitive because of the need to tune the diffusion parameters for every new scenario of interest. The adoption and utility of the diffusion process in pattern theory may be increased if these step sizes could be 1) automatically determined for each new target template and 2) independent of the target's viewing perspective and range to the imaging sensor.

This chapter considers an alternative to Langevin-based diffusions for pattern-theoretic ATR, inspired by grid search-based optimization. We discuss the mapping from real world coordinates to two-dimensional image coordinates and how this mapping can be used to define a new type of diffusion that adjusts target position by individual pixel values. This mapping is used to facilitate a search among the pose parameters within integer pixel units from the current position estimate via Metropolis-Hastings sampling. Using this scheme, adjustments can be made to the configuration parameters so that the differences between the estimated target pose parameters at each step of the iteration correspond to individual pixel sizes, potentially providing more reliable and efficient convergence with no a priori step size determination. We show that this scheme performs well for refining a target's (x, y) position parameters. For a target's orientation angle θ, visible changes in orientation from the sensor's point of view will always depend on the relative position of the target with respect to the sensor, the distance between target and sensor, and the target's viewing geometry. However, the real-world coordinate to image coordinate mapping technique will only require one step size that is specified as an angle covering the parameter search space.

3.2 Background

As illustrated in Chapter 2, some pattern-theoretic ATR algorithms depend on methods that iteratively adjust continuous target pose parameters using sensor likelihood information to refine each parameter estimate. While the Langevin stochastic differential equation has been previously used for pattern-theoretic ATR, there are many other techniques that could be applied to the same problem.

Evolutionary algorithms encompass techniques that are generally applied to continuous parameter refinement problems. Differential evolution (DE) is one such approach [59]. DE is a heuristic for achieving global optimization of continuous functions, specifically when those functions are nonlinear and non-differentiable. It also requires few user-specified convergence parameters, since the search space is altered by considering the difference of intermediate terms within the search space. DE is similar to another class of evolutionary approaches called genetic algorithms (GA). Both are inspired by the genetic processes found in biology (such as crossover and mutation) to search over and refine solutions to optimization problems [2]. For the pose parameter diffusion problem, both DE and GA suffer from initial application of their framework to the pose refinement problem and the need to choose convergence parameters that do not directly map to the scene or sensor characteristics.

3.2.1 Langevin Diffusions in Pattern-Theoretic ATR

This section reviews and expands upon our discussion from Section 2.1.4 about the Langevin SDE used to define the diffusion process for pattern-theoretic ATR. In general, SDEs are differential equations containing terms that are random processes. Therefore, the solution to the SDE must also be a random process. SDEs are important tools used to characterize a number of processes governed by seemingly random fluctuations, like the movement of particles in water or changes observed in stock prices [30, 50].

The pattern-theoretic automatic target recognition (ATR) algorithm described in Chapter 2 uses a type of jump-diffusion process to iteratively estimate the target parameters of interest for targets in a given image. These parameters include target type, position, and orientation. Synthetic scenes containing the hypothesized targets are compared to the original image using a likelihood function based on the statistics of the sensor used to acquire the original image. The jumps determine the size of the parameter space, i.e., the number of targets, while diffusions update pose parameter estimates [25, 40, 39, 13].

During the intervals between jumps, the diffusion process takes over and adjusts the continuous pose parameters $[x, y, \theta]$ by small amounts to better align the hypothesized targets with the corresponding targets in the image data.

Implementations of the diffusion process have previously used a discretized form of the Langevin SDE

$$d\mathbf{C}_N(\tau) = \nabla_{\mathbf{C}_N} H[\mathbf{C}_N(\tau)|\mathbf{d}]d\tau + \sqrt{2}dW_N, \tag{21}$$

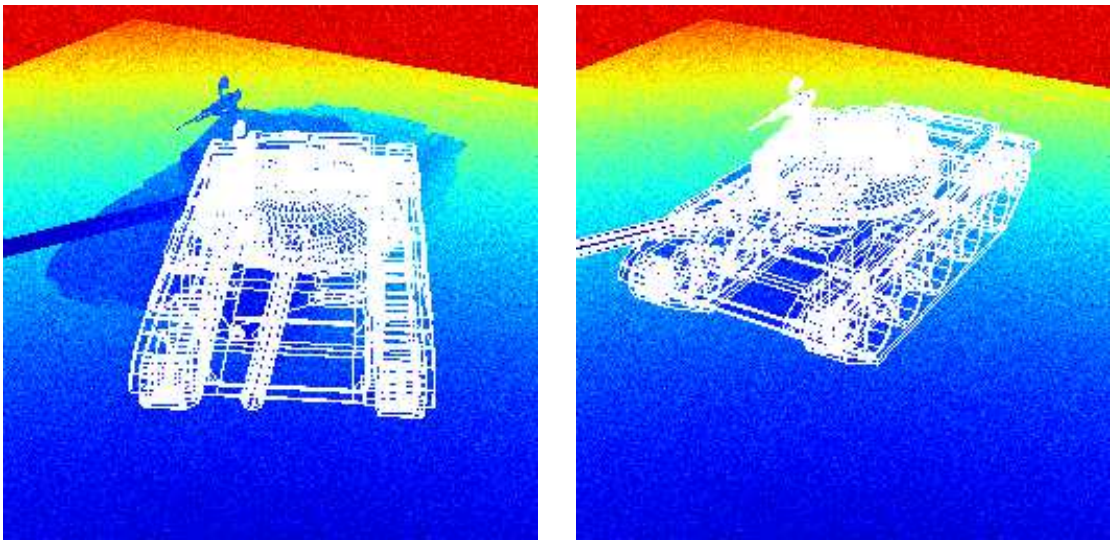
where W_N is a Wiener process and $H[\mathbf{C}_N(\tau)|\mathbf{d}]$ is the logposterior associated with the configuration parameter vector $\mathbf{C}_N$, which contains the configuration parameters for N targets of fixed classes. The time index τ refers to a unit of time within the diffusion interval. Once (21) is discretized, τ can simply be thought of as a discrete time index such that a finite number of diffusions occur between jumps and that number is an exponential random variable.

The derivative needed to compute the Langevin SDE (21) can be estimated by applying a finite difference approximation:

$$\frac{\partial H(c|\mathbf{d})}{\partial c_p} \approx \frac{H(\ldots, c_p + \delta, \ldots|\mathbf{d}) - H(\ldots, c_p - \delta, \ldots|\mathbf{d})}{2\delta}, \tag{22}$$

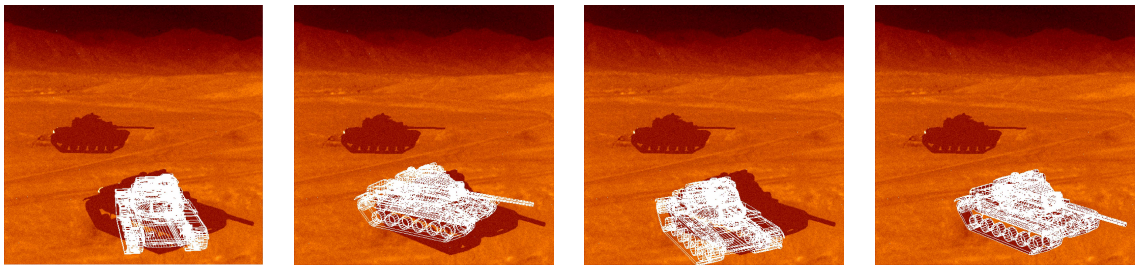
where c_p is a single parameter of the configuration c, δ is some small deviation of the parameter c_p, and the ellipses indicate the remaining parameters are held fixed.

To summarize this process, the jump-diffusion algorithm starts by estimating an approximate location for a target. If we create a scene by rendering a target at this estimated location, which we will call the hypothesis, we will see that the hypothesized target and corresponding target in the data partially overlap. To refine the initial guess, the diffusion process incrementally adjusts the pose parameters, using the information in the image data. When viewing the hypothesis rendered on top of the data, the overlap between the two improves as the estimated pose parameters converge to their true values. See Figures 6 and 7 for examples of the diffusion process.



(a) (b)

Figure 6: Images (a) and (b) contain a sample, synthetic, noisy, LADAR image. A T62 tank sits at the origin of a ground plane. If we assume that our ATR algorithm initially detects a target in the general vicinity of the T62, we may find that our hypothesized T62, denoted by the wire frame outline, does not overlap perfectly with the T62 in the data. This is shown in (a). After a few iterations of the diffusion process, the pose parameters of the hypothesized T62 should match well with those of the data T62, and the hypothesized T62 should overlap well with the data image, as shown in (b).



(a) (b) (c) (d)

Figure 7: Images (a), (b), and (c) show a hypothesized target rendered over sample FLIR image data. In each of these images, the estimated pose parameters do not match the true values for the corresponding target within the data image, but there is some overlap. After a sufficient number of iterations of the diffusion process, the estimated pose parameters will adjust until there is a closer match with the data, as shown in image (d).

3.3 Langevin-Based Diffusion Implementation Challenges

When simulating the Langevin SDE, two issues arise: the choice of step sizes for the derivative computation and the choice of step sizes for the discretized diffusions themselves. In earlier jump-diffusion experiments [40, 39], both of these were determined empirically through a trial-and-error approach that yielded the best adjustments to the configuration parameters. Ideally, these should be automatically determined, but in practice this is problematic because they depend on the types of targets in the scene, the scene's viewing parameters, and the sensor likelihood function.

Also, Langevin-style diffusions are more natural when there is an analytic expression for the likelihood derivative. As defined in our pattern-theoretic ATR implementation, this derivative must be approximated with the finite difference equation shown in (11). The logposterior H is effectively a function of an image and we are taking the derivative of this function with respect parameters used to generate the image. Considering the nature of this approximation, this demonstrates another reason why determining ideal step sizes for $d\tau$ and dW_N from (10) is not intuitive.

Lastly, diffusions of this form may lead to crude approximations to the pose of detected targets. In some cases, the Langevin diffusion may not converge, but instead

oscillate among values close to the true target pose. These characteristics are common in Langevin-style diffusions [18].

The following discussion will examine a new diffusion algorithm that redefines the pose parameter refinement problem in way that is more natural to implement and less reliant on empirically determined step sizes.

3.4 Pixel-Based Diffusion

In the new diffusion algorithm, we note that the Langevin SDE is not essential to determining appropriate changes in the coordinate pose parameters (x and y ground plane positions), because those parameters are naturally discretized by the inter-pixel spacing within the image of interest, mapping to some real-world unit of distance measurement. For example, if the spacing between neighboring pixels is a centimeter, adjusting the pose parameters by millimeters may not result in a visible change to the hypothesized image. It is also possible for a target to move by an amount that is less than the spacing between image pixels, yet still resulting in a visible change to the imaged target's intensity values. This naturally leads to the possibility of adjusting the pose parameters by amounts that are below the pixel resolution of the image created under the sensor's viewing parameters. Tools used to generate two-dimension scenes from a three-dimensional configuration, like OpenGL, may provide various ways to handle transitions across pixel boundaries (antialiasing and subpixel rendering are two such techniques) [27]. For the discussion to follow, we will limit ourselves to pixel-level pose refinement for the coordinate pose parameter pair (x, y).

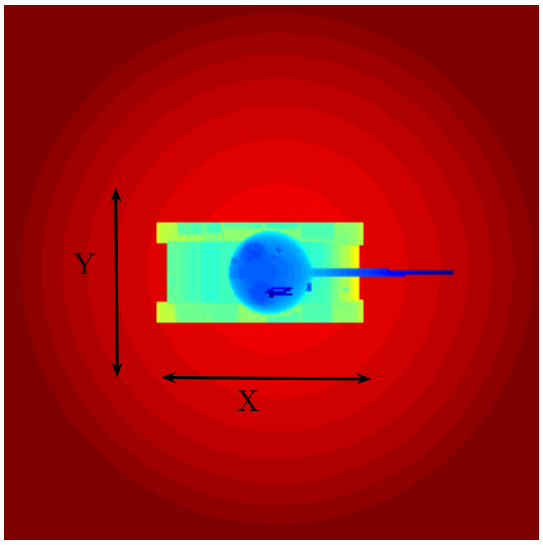
The process begins by choosing a *pixel radius*, the desired number of adjacent pixels to consider when determining an updated set of pose parameters for the hypothesized target centered at pixel origin $p_{x_0y_0}$. A larger pixel radius implies that a larger search space will be considered, increasing the computation time per iteration but decreasing the number of iterations necessary to reach good pose values. Once

a pixel radius is chosen, the algorithm proceeds by adjusting the pose parameters for the hypothesized target, selecting values that would result in the image of the hypothesized target moving by a single pixel, or multiple pixels, across the projected surface of the image (see Figure 8).

To transform the two-dimensional pixel coordinates p_{xy} to three-dimension scene coordinates $[x_0, y_0, \theta_0]$, we undo the perspective projection transformation discussed in Section 2.1.2. While the rendering processing creates a mapping between the three-dimensional scene coordinates and the two-dimensional coordinates of the resulting image, undoing the perspective projection reverses the operation. Since each two-dimensional image coordinates can represent multiple three-dimensional scene coordinates, we limit the possible three-dimensional coordinates by constraining the reverse projection to points on the same *ground plane* where the target of interest is seated.

In addition to varying the coordinate parameters, we must also consider the orientation parameter θ. This parameter represents the ground-based rotation angle of the target of interest. Rotations do not fit our pixel-based adjustment model as well as changes to the position coordinates, so we are left with choosing a method that will appropriately sample small deviations of θ. This effectively rotates the hypothesized targets by small amounts in both directions. We have found that specifying a *sweep angle* ϕ_s and *sweep angle step size* $d\phi_s$ is flexible enough to allow the algorithm to converge to pose values matching those of the corresponding target in the data reasonably well, under many varying conditions.

Once the candidate coordinate positions have been chosen, the posterior probabilities are computed for each candidate using the same Gibbs form as discussed in Section 2.1.3. To fit within the jump-diffusion framework previously established in Section 2.1.4, we would select one of these candidates probabilistically with a probability that proportional to its posterior probability. This candidate is accepted with



(a)

(-13.67, 13.67)	(-6.83, 13.67)	(0.00, 13.67)	(6.83, 13.67)	(13.67, 13.67)
(-13.67, 6.83)	(-6.83, 6.83)	(0.00, 6.83)	(6.83, 6.83)	(13.67, 6.83)
(-13.67, 0.00)	(-6.83, 0.00)	(0.00, 0.00)	(6.83, 0.00)	(13.67, 0.00)
(-13.67, -6.83)	(-6.83, -6.83)	(0.00, -6.83)	(6.83, -6.83)	(13.67, -6.83)
(-13.67, -13.67)	(-6.83, -13.67)	(0.00, -13.67)	(6.83, -13.67)	(13.67, -13.67)

(b)

Figure 8: In image (a), a tank is located at some ground-based position that we denote (0.00, 0.00). The tank is imaged by a LADAR sensor positioned 100 m away, with a 30° field-of-view, pointing directly toward the tank. If a pixel radius of two were chosen for the diffusion, the 24 pixels surrounding the origin pixel would be selected as candidates for the first iteration. For the image as specified in (a), the 24 test pixels and origin pixel will correspond to the grid of (x, y) pose parameters, in centimeters, found in (b). In this case, the space between each pixel is approximately 6.83 cm.

probability $\beta(c_{curr}, c_{next})$, calculated using a Metropolis-Hastings approach:

$$\beta(c_{curr}, c_{next}) = \min\left(\frac{\pi(c_{next})r(c_{next}, c_{curr})}{\pi(c_{curr})r(c_{curr}, c_{next})}, 1\right). \tag{23}$$

The term c_{next} is the proposed state of the configuration, while c_{curr} is the current state. The functions of the form $r(c_1, c_0)$ are transition probabilities, specifying the probability of reaching candidate state c_1 from state c_0. More details on Metropolis-Hastings in the context of pattern-theoretic ATR may be found in [40]. The function $\pi(c)$ is the probability of being in state c, which in this case is derived from the logposterior H. When selecting candidates during an iteration, candidate position and orientation values can be adjusted by adding a noise term dW_N, just as it was done in the Langevin SDE approach, to reduce the probability of becoming trapped in one of the posterior distribution's local maximums.

A summary of the diffusion algorithm is shown in Algorithm 1.

Algorithm 1 Pixel-based diffusion algorithm.

Initialize pose parameters $c_{next} \leftarrow [x_0, y_0, \theta_0]$
Initialize pixel radius r_p
Initialize sweep angle ϕ_s
Initialize step size $d\phi_s$
repeat
    Assign $c_{curr} \leftarrow c_{next}$
    Compute origin pixel coordinate $p_{x_0y_0}$ from c_{curr}
    Find the set of pixels $\mathcal{R}_p = p_{xy} : \|p_{xy} - p_{x_0y_0}\| < r_p$
    Find the $N_\phi = \frac{\phi_s}{d\phi_s}$ angular sweep steps in the set $\mathcal{N}_\phi$
    for all pose parameter coordinates in the space $\mathcal{R}_p \times \mathcal{N}_\phi$ **do**
        Compute the logposterior probability H
    end for
    Create a probability distribution from H over the space $\mathcal{R}_p \times \mathcal{N}_\phi$
    Draw c_{next} probabilistically from the H distribution to obtain proposed $[x, y, \theta]$
    Accept c_{next} with probability $\beta(c_{curr}, c_{next})$
until $c_{curr} = c_{next}$

3.5 Simulations with LADAR data

This section compares results of performing the diffusion on a test data set using both the original Langevin-based scheme and the pixel-based scheme developed in this

section. The first test considers a single M-60 tank under similar downward-looking viewing geometries, as seen in previous simulations used in this dissertation. The second scenario considers a single T-62, target but uses a lower angle of elevation to consider what happens when one or more of the pose parameters is heavily influenced by the target depth within the image. Both scenarios use simulated LADAR imagery instead of FLIR primarily to illustrate how pattern-theoretic ATR techniques can be readily applied to data from different sensor types. Future work can show that the pixel-based diffusion can be readily applied to FLIR data as well.

Since diffusions occur after a candidate target has been identified, we start the simulations with an initial pose parameter estimate that results in the hypothesized target partially overlapping the actual target in the image, as perceived by the sensor. Starting from this point, four sets of simulations are performed:

- The x parameter is unknown, while the y and θ parameters are known and fixed.
- The y parameter is unknown, while the x and θ parameters are known and fixed.
- The θ parameter is unknown, while the x and y parameters are known and fixed.
- All three parameters, x, y, and θ, are unknown and jointly estimated.

3.5.1 Case 1: A Single M-60 Tank

Figure 9 contains the LADAR image of a single M-60 tank, resting on the ground plane and pointing 135 degrees away from the imaging sensor. The wireframe image of the tank represents the rendering of the tank based on the initial coarse estimate of the pose parameters, typical of those that may be obtained from a birth or metamorph event in a jump diffusion ATR scenario.

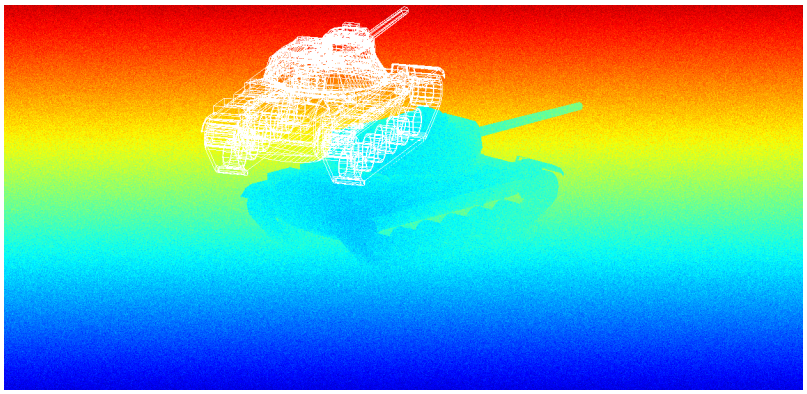


Figure 9: A synthetic LADAR image containing a single M-60 tank. The wireframe overlay represents a typical starting point for the diffusion process.

Figure 10 shows the results of applying Algorithm 1 to the scene in Figure 9. The plots show the deviation of each estimated parameter from the true value for that parameter during iterations of the Langevin diffusion and the pixel-based diffusion. From these plots, we can observe the following:

- The pixel-based diffusion algorithm converges faster than the Langevin SDE algorithm.
- The convergence curve of the pixel-based approach usually appears monotonic. In the Langevin diffusion process, we see "jitter" in the pose parameter estimates, where it appears the pose parameter oscillates around some value that it can never quite obtain.

Interestingly, the greatest difference between the two diffusion methods is seen in Figure 10(c). The diffusion of the target's orientation angle converges faster than the other parameters when using the pixel-based diffusion algorithm, yet it converges slower than the other parameters when using the Langevin-based algorithm. We observe that small changes in target orientation angle tend to produce fewer visual differences than changes in target coordinate positions that cross pixel boundaries. This may not affect the pixel-based algorithm as much as the Langevin-based diffusion since the former can directly sample from candidates at small orientation angles. The

Langevin-based algorithm must compute these values, which can be sensitive to the diffusion and derivative step sizes.

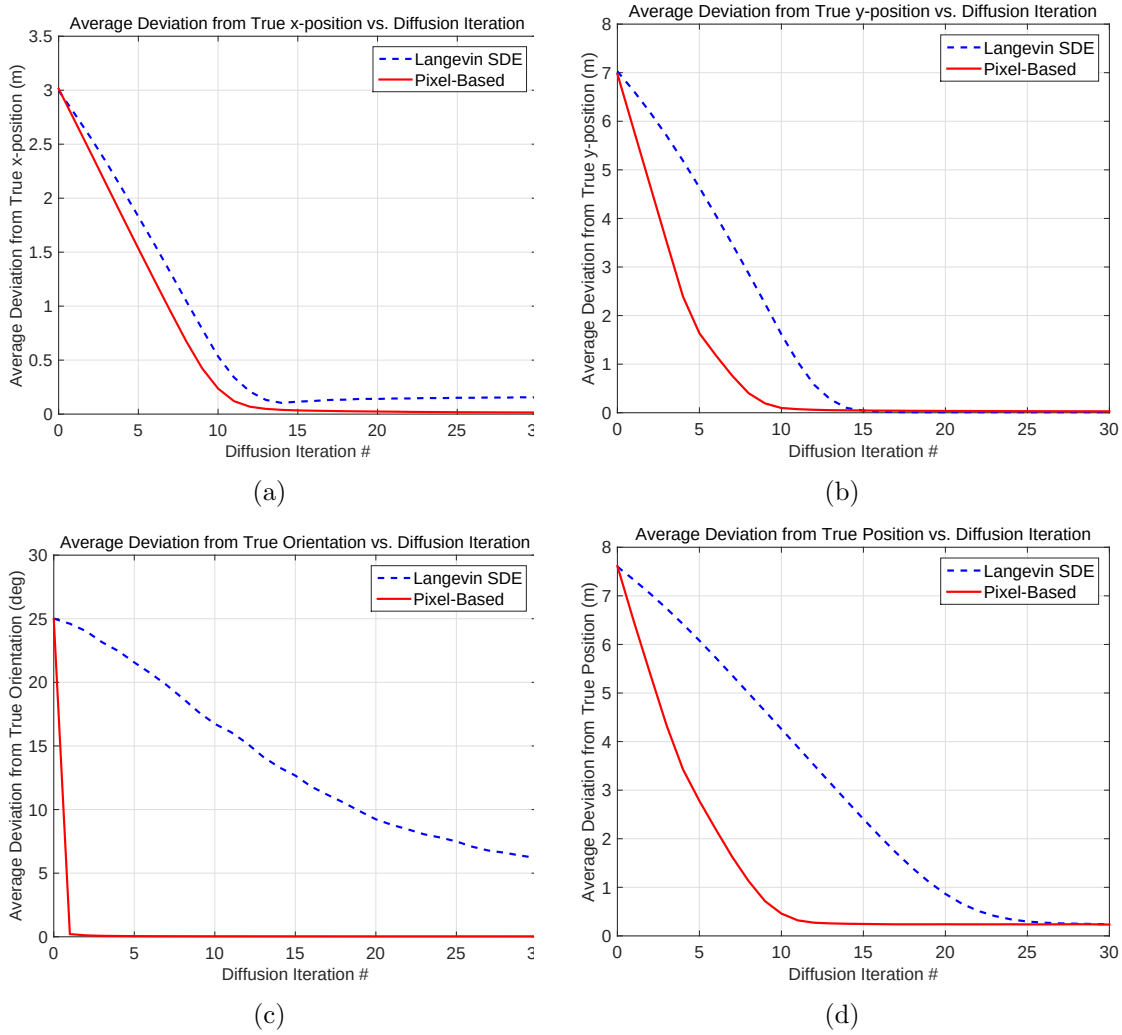


Figure 10: Charts showing parameter deviation vs iteration index for Langevin diffusions and pixel-based diffusions for a single M-60 tank averaged over 1000 trials. Figure 10(a) shows the deviation of x, in meters, when y and θ are held constant. Figure 10(b) shows the deviation of y, in meters, when x and θ are held constant. Figure 10(c) shows the deviation of θ, in degrees, when x and y are held constant. Figure 10(d) shows the norm of the deviation of both x and y, in meters, when all parameters are unknown from the beginning.

3.5.2 Case 2: A Single T-62 Tank

Figure 11 contains the LADAR image of a single T-62 tank, resting on the ground plan and pointing 90 degrees away from the imaging sensor. Compared to Case 1,

the imaging sensor has a lower elevation angle. This results in a scene only showing a single side of the target at any given orientation angle. Unlike the target geometry viewed in Case 1, scenes generated with different candidate orientation angles will visually have less in common with each other.

Figure 11: A synthetic LADAR image of a single T-62 tank. The wireframe overlay represents a typical starting point for the diffusion process.

The results of the simulation can be seen in Figure 12. Just as in Case 1, the pixel-based diffusions converge faster than the Langevin-based diffusions. The reduced elevation angle slightly affects the convergence with the Langevin SDE, but not the pixed-based diffusion. The differences between the two algorithms are even more apparent when looking at the results of diffusing the orientation parameter. The lower elevation angle results in worse performance in the Langevin-based approach.

3.6 Conclusion

This chapter discussed diffusion algorithms as applied to the pattern-theoretic ATR problem. We described the previously used Langevin-based approach and its shortcomings while presenting a pixel-based alternative. We simulated each diffusion under a number of test conditions to compare the convergence of both techniques. We found that the pixel-based approach offers a natural implementation and does not rely on

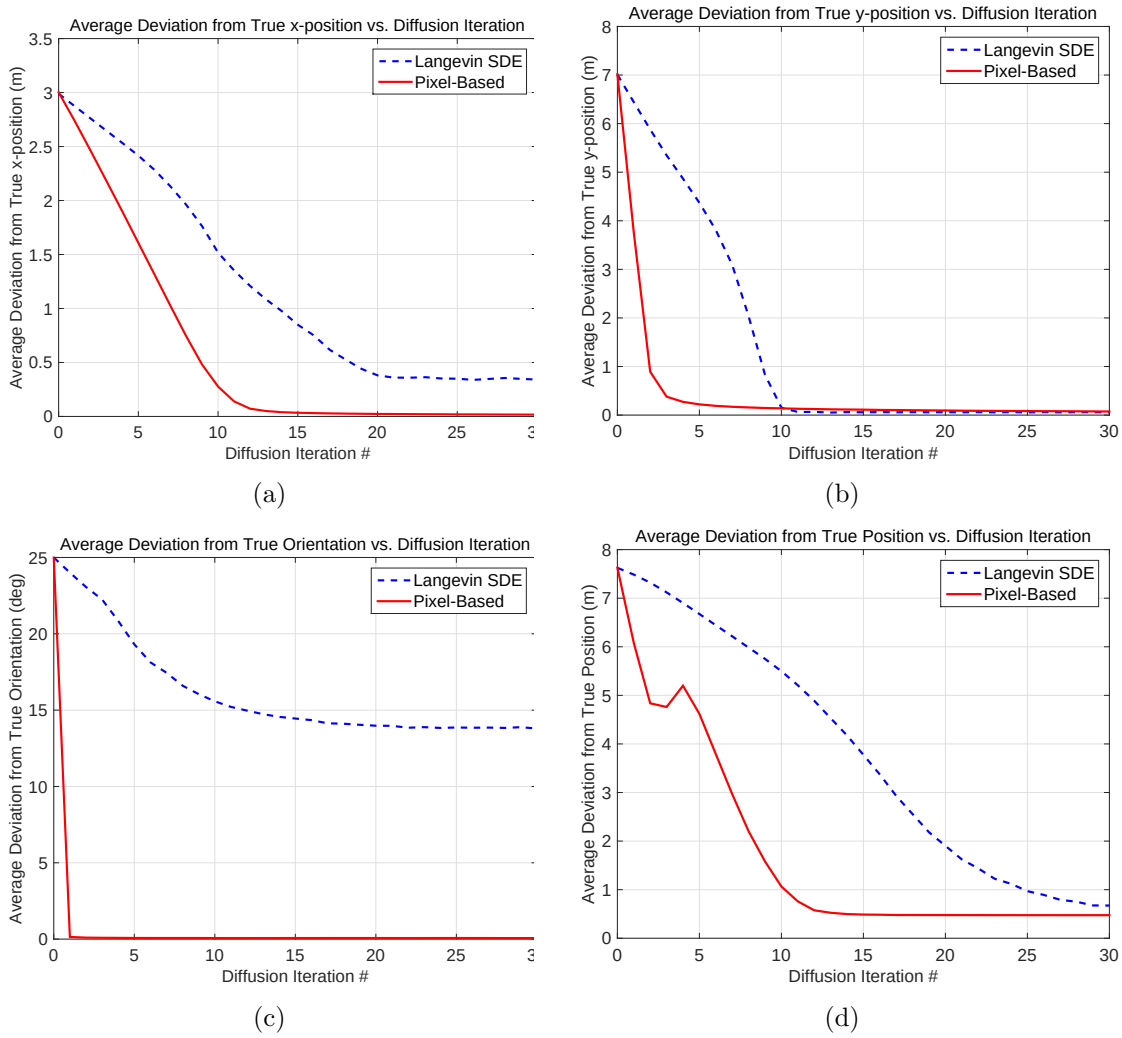


Figure 12: Charts showing parameter deviation vs iteration index for langevin diffusions and pixel-based diffusions for a single T-62 tank averaged over 1000 trials. Figure 12(a) shows the deviation of x, in meters, when y and θ are held constant. Figure 12(b) shows the deviation of y, in meters, when x and θ are held constant. Figure 12(c) shows the deviation of θ, in degrees, when x and y are held constant. Figure 12(d) shows the norm of the deviation of both x and y, in meters, when all parameters are unknown from the beginning.

empirically determined step sizes that may need to change with scene viewing geometry. The orientation parameter is still slightly problematic since there are infinite angular steps to test against, even when considering the constraint of a two-dimensional image plane.

The pixel-based diffusion can naturally replace the Langevin in the full jump-diffusion ATR algorithm, and its implementation is an avenue for future work.

CHAPTER IV

THERMAL STATE INFERENCE WITH INFRARED CALIBRATION

4.1 Introduction

The techniques for pattern-theoretic ATR described in [40, 41] were evaluated on synthetic infrared data sets, created directly from known thermal intensity values. This was appropriate for initial tests of the effectiveness of pattern-theoretic ATR, but it assumed that the infrared sensor is calibrated so that there exists a known mapping from thermal intensities to digitized sensor units. This chapter discusses techniques for handling of uncalibrated infrared data under the pattern-theoretic framework. In particular, we address the problem of jointly estimating the parameters that determine the thermal characteristics of a target of interest along with the terms that govern the calibration of measured sensor units to thermal state.

4.2 Background

This dissertation uses the term "calibration" differently than how it is usually used in computer vision. In the context of computer vision, camera calibration typically refers to the process of determining intrinsic parameters of a camera using a set of ground truth images with known metrics. The purpose of calibration is to determine, then correct, camera deficiencies related to parameters like focal length, pixel size, radial distortion. This dissertation considers calibration as it relates to the mapping between infrared sensor units and the corresponding thermal state of the target within the sensor's field of view. We only note the computer vision definition to highlight the differences between the two problems.

Sensor calibration is a common problem spanning many disciplines. For any sensor that digitally captures real-world data, a mapping between the units of the sensor's analog-to-digital converter to actual physical units is necessary to accurately determine the conditions under which the data was recorded. In thermal imagining, a sensor tells us in relative terms what parts of an imaged object are "hotter" than others, but calibration seeks the absolute temperature of different parts of the imaged object. For pattern-theoretic approaches to the infrared ATR problem, the determination of this mapping is extremely important. Since image synthesis is the core of pattern-theoretic recognition algorithms, the units of the models used to synthesize intermediate and final target imagery must match those of the data collected. This information is not always available a priori and can change with sensor viewing geometry.

Standard infrared calibration procedures take many variables into account, like atmospheric conditions, the time of day of sensor measurement, and the viewing geometry with respect to the target surface [31, 32]. We assume these parameters are tied to a known low-dimensional thermal representation, and focus on the connection between the sensor readings and this model. The following sections treat the calibration problem like a general parameter estimation problem, seeking the MAP estimate for the thermal intensity for a collection of targets given the sensor-pixel levels and a low-dimensional representation of the thermal characteristics of targets.

4.3 The Sensor Likelihood Considering Calibration Terms

We assume that the relationship between the model and the image is affine and can be summarized as

$$\Lambda_{i,t} = a\lambda_{i,t} + b \tag{24}$$

where $\Lambda_{i,t}$ is the intensity for target t at intensity region i in the image acquired by the FLIR sensor, a and b are the calibration coefficients, and $\lambda_{i,t} = \sum_j \alpha_{j,t}\Phi_{ij,t} +$

$m_{i,t}$, as defined previously in Section 2.2.1, except that we now consider an index t for a specific target in a given scene. We will also define a new variable $\boldsymbol{\omega} = (\alpha_{1,1}, \cdots, \alpha_{J_T,T}, a, b)$ to represent the collection of all thermal expansion and calibration parameters. The technique for estimating expansion coefficients presented previously can be expanded to include the additional calibration terms. This creates a nonlinear relationship among the parameters, so a new solution must be derived. A novel iterative procedure is used to address this problem. The analysis follows the previous derivation for the logposterior for pixels on target in terms of the expansion coefficients defined in Section 2.2.2, except we now include the calibration coefficients a and b as well. This new logposterior may be written as

$$H(\boldsymbol{\omega}|D) = -\sum_t \sum_i \sum_{k \in R_{i,t}} \frac{(\Lambda_{i,t} - d(k))^2}{2\sigma^2} - \sum_t \sum_j \frac{\alpha_{j,t}^2}{2\gamma_{j,t}} \tag{25}$$

$$= -\sum_t \sum_i \sum_{k \in R_{i,t}} \frac{\Lambda_{i,t}^2 - 2\Lambda_{i,t} d(k) + d^2(k)}{2\sigma^2} - \sum_t \sum_j \frac{\alpha_{j,t}^2}{2\gamma_{j,t}} \tag{26}$$

$$= -\sum_t \sum_i \frac{N_{i,t}\Lambda_{i,t}^2 - 2\Lambda_{i,t} \sum_{k \in R_{i,t}} d(k) + \sum_{k \in R_{i,t}} d^2(k)}{2\sigma^2} - \sum_t \sum_j \frac{\alpha_{j,t}^2}{2\gamma_{j,t}} \tag{27}$$

$$= -\sum_t \sum_i \frac{N_{i,t}\Lambda_{i,t}^2 - 2\Lambda_{i,t} D_{i,t} + D_{i,t}}{2\sigma^2} - \sum_t \sum_j \frac{\alpha_{j,t}^2}{2\gamma_{j,t}}. \tag{28}$$

4.4 Estimating the Calibration and Thermal Parameters

Incorporating $\Lambda_{i,t} = a\lambda_{i,t} + b$, where $\lambda_{i,t} = \sum_j \alpha_{j,t}\Phi_{ij,t} + m_{i,t}$, we must take the derivatives of $H(\boldsymbol{\omega}|D)$ with respect to each $\alpha_{j,t}$, a, and b. To make the following

derivations cleaner, we mention the following derivatives:

$$\frac{\partial}{\partial \alpha_{j,t}} \Lambda_{i,t'} = a \frac{\partial}{\partial \alpha_{j,t}} \lambda_{i,t'} = \begin{cases} a\Phi_{ij,t} & \text{if } t = t' \\ 0 & \text{if } t \neq t' \end{cases}, \tag{29}$$

$$\frac{\partial}{\partial a} \Lambda_{i,t} = \lambda_{i,t}, \tag{30}$$

$$\frac{\partial}{\partial b} \Lambda_{i,t} = 1. \tag{31}$$

The derivatives of $H(\boldsymbol{\omega}|D)$ with respect to each $\alpha_{j,t}$ are:

$$\frac{\partial H}{\partial \alpha_{j,t}} = -\sum_{t'} \sum_{i} \frac{2N_{i,t'} \Lambda_{i,t'} \frac{\partial}{\partial \alpha_{j,t}} \Lambda_{i,t'} - 2 \frac{\partial}{\partial \alpha_{j,t'}} \Lambda_{i,t'} D_{i,t'}}{2\sigma^2} - \frac{\alpha_{j,t}}{\gamma_{j,t}} \tag{32}$$

$$= -\sum_{i} \frac{N_{i,t} \left[a \left(\sum_k \alpha_{k,t} \Phi_{ik,t} + m_{i,t} \right) + b \right] a\Phi_{ij,t} - a\Phi_{ij,t} D_{i,t}}{\sigma^2} - \frac{\alpha_{j,t}}{\gamma_{j,t}} \tag{33}$$

$$= -\sum_{i} \frac{\left(aN_{i,t} \sum_k \alpha_{k,t} \Phi_{ik,t} + aN_{i,t} m_{i,t} + N_{i,t} b \right) a\Phi_{ij,t} - a\Phi_{ij,t} D_{i,t}}{\sigma^2} - \frac{\alpha_{j,t}}{\gamma_{j,t}} \tag{34}$$

$$= -\frac{\sum_i a^2 \Phi_{ij,t} N_{i,t} \sum_k \alpha_{k,t} \Phi_{ik,t} + \sum_i \Phi_{ij,t} \left(a^2 N_{i,t} m_{i,t} + abN_{i,t} - aD_{i,t} \right)}{\sigma^2} - \frac{\alpha_{j,t}}{\gamma_{j,t}} \tag{35}$$

$$= -\frac{a^2}{\sigma^2} \sum_k \alpha_{k,t} \sum_i N_{i,t} \Phi_{ik,t} \Phi_{ij,t} + \frac{a^2}{\sigma^2} \sum_i \Phi_{ij,t} \left(N_{i,t} m_{i,t} + \frac{bN_{i,t}}{a} - \frac{D_{i,t}}{a} \right) - \frac{\alpha_{j,t}}{\gamma_{j,t}}. \tag{36}$$

To maximize the logposterior with respect to the $\alpha_{j,t}$ terms, we must satisfy these $\sum_t J_t$ necessary conditions, which were obtained by setting $\frac{\partial H}{\partial \alpha_{j,t}} = 0$ and dividing each term by $-\frac{a^2}{\sigma^2}$:

$$\sum_k \alpha_{k,t} \sum_i N_{i,t} \Phi_{ik,t} \Phi_{ij,t} - \sum_i \Phi_{ij,t} \left(N_{i,t} m_{i,t} + \frac{bN_{i,t}}{a} - \frac{D_{i,t}}{a} \right) + \frac{\sigma^2 \alpha_{j,t}}{a^2 \gamma_{j,t}} = 0, \ \forall j, t. \tag{37}$$

The derivative of $H(\boldsymbol{\omega}|D)$ with respect to a is

$$\frac{\partial H}{\partial a} = -\sum_t \sum_i \frac{2N_{i,t}\Lambda_{i,t}\frac{\partial}{\partial a}\Lambda_{i,t} - 2\frac{\partial}{\partial a}\Lambda_{i,t}D_{i,t}}{2\sigma^2} \tag{38}$$

$$= -\sum_t \sum_i \frac{N_{i,t}\Lambda_{i,t}\lambda_{i,t} - \lambda_{i,t}D_{i,t}}{\sigma^2} \tag{39}$$

$$= -\sum_t \sum_i \frac{N_{i,t}\left(a\lambda_{i,t}+b\right)\lambda_{i,t} - \lambda_{i,t}D_{i,t}}{\sigma^2} \tag{40}$$

$$= -\sum_t \sum_i \frac{aN_{i,t}\lambda_{i,t}^2 + bN_{i,t}\lambda_{i,t} - \lambda_{i,t}D_{i,t}}{\sigma^2} \tag{41}$$

$$= -\frac{a}{\sigma^2}\sum_t \sum_i N_{i,t}\lambda_{i,t}^2 - \frac{b}{\sigma^2}\sum_t \sum_i N_{i,t}\lambda_{i,t} + \frac{1}{\sigma^2}\sum_t \sum_i \lambda_{i,t}D_{i,t}, \tag{42}$$

and the derivative of $H(\boldsymbol{\omega}|D)$ with respect to b is

$$\frac{\partial H}{\partial b} = -\sum_t \sum_i \frac{2N_{i,t}\Lambda_{i,t}\frac{\partial}{\partial b}\Lambda_{i,t} - 2\frac{\partial}{\partial b}\Lambda_{i,t}D_{i,t}}{2\sigma^2} \tag{43}$$

$$= -\sum_t \sum_i \frac{N_{i,t}\Lambda_{i,t} - D_{i,t}}{\sigma^2} \tag{44}$$

$$= -\sum_t \sum_i \frac{N_{i,t}\left(a\lambda_{i,t}+b\right) - D_{i,t}}{\sigma^2} \tag{45}$$

$$= -\sum_t \sum_i \frac{aN_{i,t}\lambda_{i,t} + bN_{i,t} - D_{i,t}}{\sigma^2} \tag{46}$$

$$= -\frac{a}{\sigma^2}\sum_t \sum_i N_{i,t}\lambda_{i,t} + \frac{b}{\sigma^2}\sum_t \sum_i N_{i,t} - \frac{1}{\sigma^2}\sum_t \sum_i D_{i,t}. \tag{47}$$

To maximize this part of the posterior, the derivatives must satisfy

$$\begin{aligned} a\sum_t \sum_i N_{i,t}\lambda_{i,t}^2 + b\sum_t \sum_i N_{i,t}\lambda_{i,t} &= \sum_t \sum_i \lambda_{i,t}D_{i,t}, \\ a\sum_t \sum_i N_{i,t}\lambda_{i,t} + b\sum_t \sum_i N_{i,t} &= \sum_t \sum_i D_{i,t}. \end{aligned} \tag{48}$$

4.5 Levenberg-Marquardt Solution to the Nonlinear System

This section describes an approximate solution to the set of equations defined in (37) and (48) using the Levenberg-Marquardt algorithm, a well-known modification to the Gauss-Newton method for solving systems of nonlinear equations [43, 46]. It can be directly applied to solving our calibration problem and while also boasting

adjustable convergence properties. To use Newton's method-style algorithms, we set up $\sum_t J_t + 2$-vector of equations in the form $\mathbf{f}(\boldsymbol{\omega}) = \mathbf{0}$ using the $\sum_t J_t$ equations from (37) and the two equations from (48). If we assume there are T targets, where target t has an eigentank decomposition consisting of J_t expansion coefficients, and $\boldsymbol{\omega}$ is the collection of FLIR thermal expansion and calibration parameters of interest, as defined in Section 4.3, then the vector $\mathbf{f}(\boldsymbol{\omega})$ would appear as follows:

$$\mathbf{f}(\boldsymbol{\omega}) = [f_{1,1}(\boldsymbol{\omega}), \cdots, f_{J_1,1}(\boldsymbol{\omega}), f_{1,2}(\boldsymbol{\omega}), \cdots, f_{J_2,2}(\boldsymbol{\omega}), \cdots, f_{J_T,T}(\boldsymbol{\omega}), f_a(\boldsymbol{\omega}), f_b(\boldsymbol{\omega})]^\top \tag{49}$$

$$= \begin{bmatrix} \sum_k \alpha_{k,1} \sum_i N_{i,1} \Phi_{ik,1} \Phi_{i1,t} - \sum_i \Phi_{i1,1} \left(N_{i,1} m_{i,1} + \frac{bN_{i,1}}{a} - \frac{D_{i,1}}{a} \right) + \frac{\sigma^2 \alpha_{1,1}}{a^2 \gamma_{1,1}} \\ \vdots \\ \sum_k \alpha_{k,1} \sum_i N_{i,1} \Phi_{ik,1} \Phi_{iJ_1,1} - \sum_i \Phi_{iJ_1,1} \left(N_{i,1} m_{i,1} + \frac{bN_{i,1}}{a} - \frac{D_{i,1}}{a} \right) + \frac{\sigma^2 \alpha_{J_1,1}}{a^2 \gamma_{J_1,1}} \\ \sum_k \alpha_{k,2} \sum_i N_{i,2} \Phi_{ik,2} \Phi_{i1,2} - \sum_i \Phi_{i1,2} \left(N_{i,2} m_{i,2} + \frac{bN_{i,2}}{a} - \frac{D_{i,2}}{a} \right) + \frac{\sigma^2 \alpha_{1,2}}{a^2 \gamma_{1,2}} \\ \vdots \\ \sum_k \alpha_{k,2} \sum_i N_{i,2} \Phi_{ik,2} \Phi_{iJ_2,2} - \sum_i \Phi_{iJ_2,2} \left(N_{i,2} m_{i,2} + \frac{bN_{i,2}}{a} - \frac{D_{i,2}}{a} \right) + \frac{\sigma^2 \alpha_{J_2,2}}{a^2 \gamma_{J_2,2}} \\ \vdots \\ \vdots \\ \sum_k \alpha_{k,T} \sum_i N_{i,T} \Phi_{ik,T} \Phi_{iJ_T,T} - \sum_i \Phi_{iJ_T,T} \left(N_{i,T} m_{i,T} + \frac{bN_{i,T}}{a} - \frac{D_{i,T}}{a} \right) + \frac{\sigma^2 \alpha_{J_T,T}}{a^2 \gamma_{J_T,T}} \\ a \sum_t \sum_i N_{i,t} \lambda_{i,t}^2 + b \sum_t \sum_i N_{i,t} \lambda_{i,t} - \sum_t \sum_i \lambda_{i,t} D_{i,t} \\ a \sum_t \sum_i N_{i,t} \lambda_{i,t} + b \sum_t \sum_i N_{i,t} - \sum_t \sum_i D_{i,t} \end{bmatrix}. \tag{50}$$

Each row of this matrix represents a single function of the form $f_\omega(\boldsymbol{\omega})$, where ω is one of the FLIR calibration parameters. The Jacobian matrix $\mathbf{J}(\boldsymbol{\omega})$ is the $\sum_t J_t + 2 \times \sum_t J_t + 2$ matrix of partial derivatives for each function $f_\omega(\boldsymbol{\omega})$ with respect to each variable of $\boldsymbol{\omega}$. It has the form

$$\mathbf{J}(\boldsymbol{\omega}) = \begin{bmatrix} \frac{\partial f_1}{\partial \omega_1} & \cdots & \frac{\partial f_1}{\partial \omega_n} \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial \omega_1} & \cdots & \frac{\partial f_n}{\partial \omega_n} \end{bmatrix}, \tag{51}$$

where $n = \sum_t J_t + 2$. We can form a matrix with this structure by using the following partial derivatives of $f_\omega(\boldsymbol{\omega})$:

$$\frac{\partial f_{j',t'}}{\partial \alpha_{j,t}} = \begin{cases} \sum_i N_{i,t}\Phi_{ij,t}^2 + \frac{\sigma^2}{a^2\gamma_{j,t}} & \text{if } j = j' \text{ and } t = t' \\ \sum_i N_{i,t}\Phi_{ij,t}\Phi_{ij',t} & \text{if } j \neq j' \text{ and } t = t' \\ 0 & \text{if } t \neq t' \end{cases} \tag{52}$$

$$\frac{\partial f_{j,t}}{\partial a} = \sum_i \Phi_{ij,t}\left(\frac{bN_{i,t} - D_{i,t}}{a^2}\right) - \frac{2\sigma^2\alpha_{j,t}}{a^3\gamma_{j,t}} \tag{53}$$

$$\frac{\partial f_{j,t}}{\partial b} = -\sum_i \frac{\Phi_{ij,t}N_{i,t}}{a} \tag{54}$$

$$\frac{\partial f_a}{\partial \alpha_{j,t}} = 2a\sum_i N_{i,t}\lambda_{i,t}\Phi_{ij,t} + b\sum_i N_{i,t}\Phi_{ij,t} - \sum_i \Phi_{ij,t}D_{i,t} \tag{55}$$

$$\frac{\partial f_a}{\partial a} = \sum_t\sum_i N_{i,t}\lambda_{i,t}^2 \tag{56}$$

$$\frac{\partial f_a}{\partial b} = \sum_t\sum_i N_{i,t}\lambda_{i,t} \tag{57}$$

$$\frac{\partial f_b}{\partial \alpha_{j,t}} = a\sum_i N_{i,t}\Phi_{ij,t} \tag{58}$$

$$\frac{\partial f_b}{\partial a} = \sum_t\sum_i N_{i,t}\lambda_{i,t} \tag{59}$$

$$\frac{\partial f_b}{\partial b} = \sum_t\sum_i N_{i,t}. \tag{60}$$

Additionally, the Levenberg-Marquardt algorithm requires setting a user defined parameter τ, which affects the convergence rate. The value chosen for τ is based on how close our initial parameter values $\boldsymbol{\omega}_0$ are to their final values. Using the function

vector $\mathbf{f}$, the Jacobian matrix $\mathbf{J}(\boldsymbol{\omega})$, a set of starting values for $\boldsymbol{\omega}_0$, our choice of τ, and two boundary terms ε_1 and ε_2, we can iteratively solve for $\boldsymbol{\omega}$ by implementing the steps shown in Algorithm 2 [45].

Algorithm 2 Levenberg-Marquardt Algorithm to compute the thermal expansion and calibration coefficients in the vector $\boldsymbol{\omega}$.

$S \leftarrow$ `false`
$n \leftarrow 0$
$\nu \leftarrow 2$
$\boldsymbol{\omega} \leftarrow \boldsymbol{\omega}_0$
$\mathbf{A} \leftarrow \mathbf{J}(\boldsymbol{\omega})^\top \mathbf{J}(\boldsymbol{\omega})$
$\mathbf{g} \leftarrow \mathbf{J}(\boldsymbol{\omega})^\top \mathbf{f}(\boldsymbol{\omega})$
$\xi \leftarrow \tau \times \max\{a_{ii}\}$
repeat
    $n \leftarrow n + 1$
    Solve $(\mathbf{A} + \xi\mathbf{I})\mathbf{h} = -\mathbf{g}$ for $\mathbf{h}$
    if $\|\mathbf{h}\| \leq \varepsilon_2(\|\boldsymbol{\omega}\| + \varepsilon_2)$ **then**
        $S \leftarrow$ `true`
    else
        $\boldsymbol{\omega}_{\text{new}} \leftarrow \boldsymbol{\omega} + \mathbf{h}$
        $F(\boldsymbol{\omega}) \leftarrow \frac{1}{2}\mathbf{f}(\boldsymbol{\omega})^\top \mathbf{f}(\boldsymbol{\omega})$
        $F(\boldsymbol{\omega}_{\text{new}}) \leftarrow \mathbf{f}(\boldsymbol{\omega}_{\text{new}})^\top \mathbf{f}(\boldsymbol{\omega}_{\text{new}})$
        $\varrho \leftarrow \frac{F(\boldsymbol{\omega}) - F(\boldsymbol{\omega}_{\text{new}})}{\frac{1}{2}\mathbf{h}^\top(\xi\mathbf{h} - \mathbf{g})}$
        if $\varrho > 0$ **then**
            $\boldsymbol{\omega} \leftarrow \boldsymbol{\omega}_{\text{new}}$
            $\mathbf{A} \leftarrow \mathbf{J}(\boldsymbol{\omega})^\top \mathbf{J}(\boldsymbol{\omega})$
            $\mathbf{g} \leftarrow \mathbf{J}(\boldsymbol{\omega})^\top \mathbf{f}(\boldsymbol{\omega})$
            $S \leftarrow (\|\mathbf{g}\|_\infty \leq \varepsilon_1)$
            $\xi \leftarrow \xi \times \max\{\frac{1}{3}, 1 - (2\varrho - 1)^3\}$
            $\nu \leftarrow 2$
        else
            $\xi \leftarrow \xi \times \nu$
            $\nu \leftarrow 2 \times \nu$
        end if
    end if
until $(S =$ `true`$)$ or $(n \geq n_{max})$

4.6 Simulations of the Levenberg-Marquardt Solution

This section presents results from experiments where the FLIR sensor calibration terms a and b and the expansion coefficients $\alpha_{j,t}$ were jointly estimated using the

Levenberg-Marquardt algorithm. We considered the following three test cases: a scene with a single M-60, another scene with a single T-62, and a final scene with both an M-60 and a T-62. We will assume we are using FLIR sensors that map the sensed thermal radiation values to detector units in the integer range $[0, 4095]$, essentially simulating a detector with 12 bits of precision.

4.6.1 Case 1: A Single M-60

A synthetic scene with a single M-60 tank is placed on a flat ground plane. The thermal profile of the tank is generated using a randomly defined set of expansion coefficients, derived from the principal components analysis used to generate the eigentank terms. The image is corrupted by Gaussian noise, and converted to 12-bit detector units. The final image is shown in Figure 13.

At each step in the Levenberg-Marquardt iteration, the norm is computed between the estimated thermal state $\Lambda_{i,t}$ and the ground-truth thermal state. Figure 14 shows the convergence of the data estimators $\Lambda_{i,t}$ to their final values. Figure 14(a) shows how the estimated thermal state at each step of the iteration compares to the ground-truth thermal state. Figure 14(b) shows how the estimate of the expansion coefficients at each step of the iteration compare to the values from the previous step.

Figure 15 shows the convergence of the calibration terms a and b. Unlike the previous figure, we only show one set of plots for each calibration term, where each plot represents the absolute difference between the estimated parameters and the ground-truth values used to generate the data set.

The Levenberg-Marquardt algorithm calls for the iteration to terminate when a set of predefined conditions are satisfied. After the iteration is complete, we use the estimated expansion and calibration coefficients to generate our hypothesized target image. The plot in Figure 16 compare the intensities from the original data and estimated target profile at each facet index, showing that the Levenberg-Marquardt

algorithm as applied here results in a hypothesized scene where the individual facet models do not exactly match the original data. However, a trend can be seen where it appears that the thermal intensity estimates attempt to move in the direction of the ground-truth thermal intensity values.

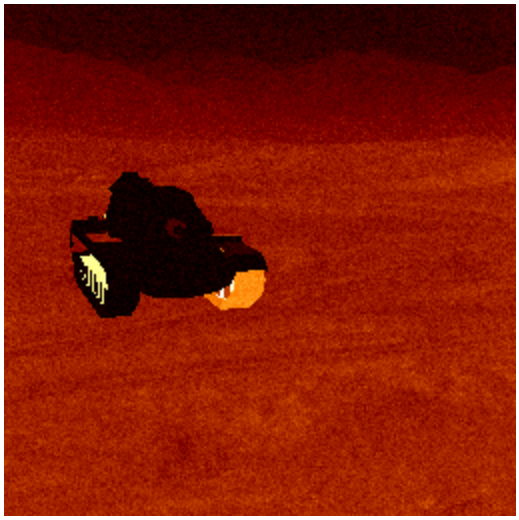
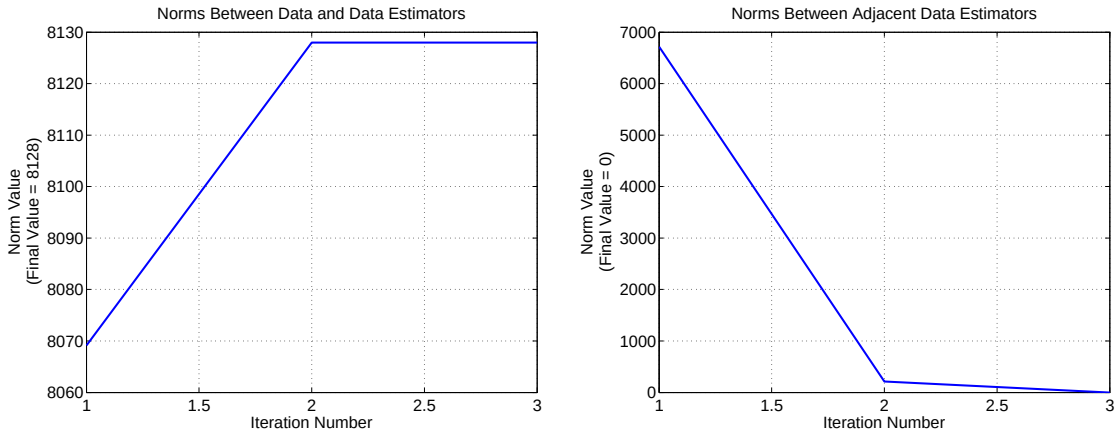


Figure 13: Scene used to test the joint estimation of eigentank expansion coefficients and FLIR calibration parameters of a single M-60 tank.

4.6.2 Case 2: A Single T-62

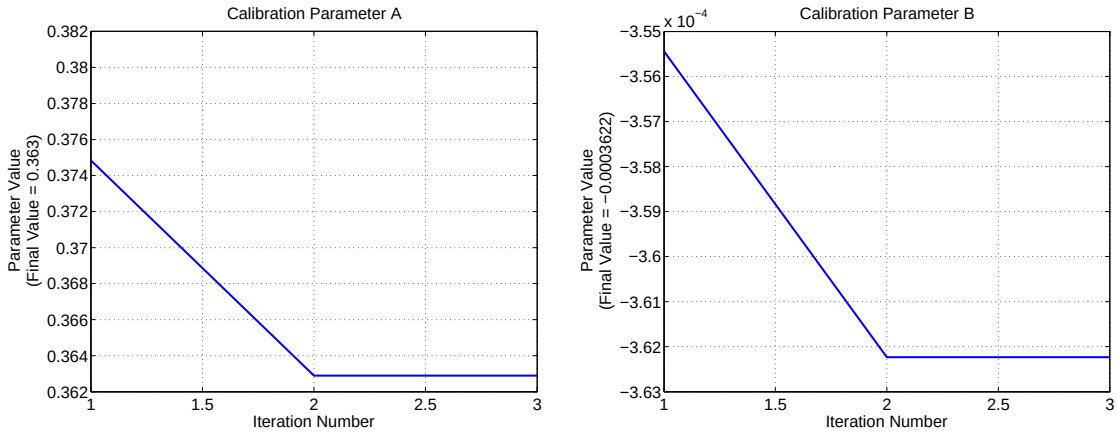
A synthetic scene with a single T-62 tank is placed on a flat ground plane in the same position as the M-60 from the previous section, as seen in Figure 17. All simulation parameters are the same as those specified in Sections 4.6 and 4.6.1.

Figures 18(a) and 18(b) and are analogous to Figures 14(a) and 14(b) from Section 4.6.1. Figure 19 is analogous to Figure 15. Though the T-62 tank is in different position and is exposing a different set of facets, thus exhibiting a different thermal state profile, the Levenberg-Marquardt algorithm continues to have difficulty obtaining an estimate that closely matches the original data, but the results look better than the M-60 case. It is even clearer that the estimated values attempt to reach the ground truth for the target.



(a) (b)

Figure 14: Figures (a) and (b) show how the M-60 estimated calibrated data changes with each iteration of the Levenberg-Marquardt algorithm. Figure (a) shows the norm between the estimated data values and the ground truth for each facet of the M-60 tank. Figure (b) shows the norm of the first difference between estimated data values.



(a) (b)

Figure 15: Figures (a) and (b) show how the M-60 calibration terms change with each iteration of the Levenberg-Marquardt algorithm. Figure (a) shows the absolute difference between the estimated and ground-truth calibration term a, while Figure (b) shows the absolute difference between the estimate and ground-truth calibration term b.

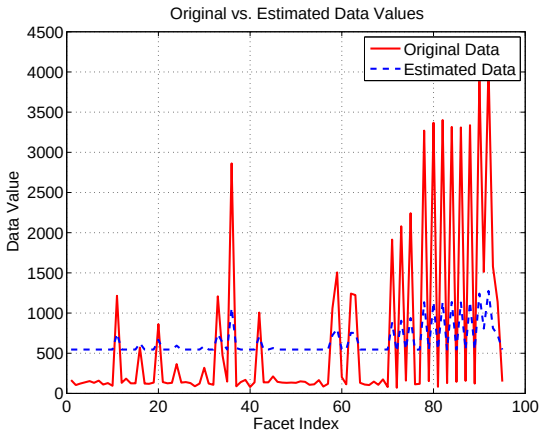


Figure 16: Comparison of the original thermal data and the thermal estimates at each facet index of an M-60 tank, obtained via the Levenberg-Marquardt algorithm. Facet indices are not listed in any particular order.

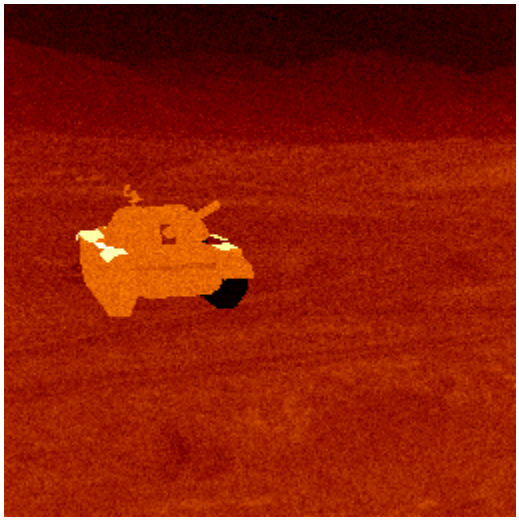
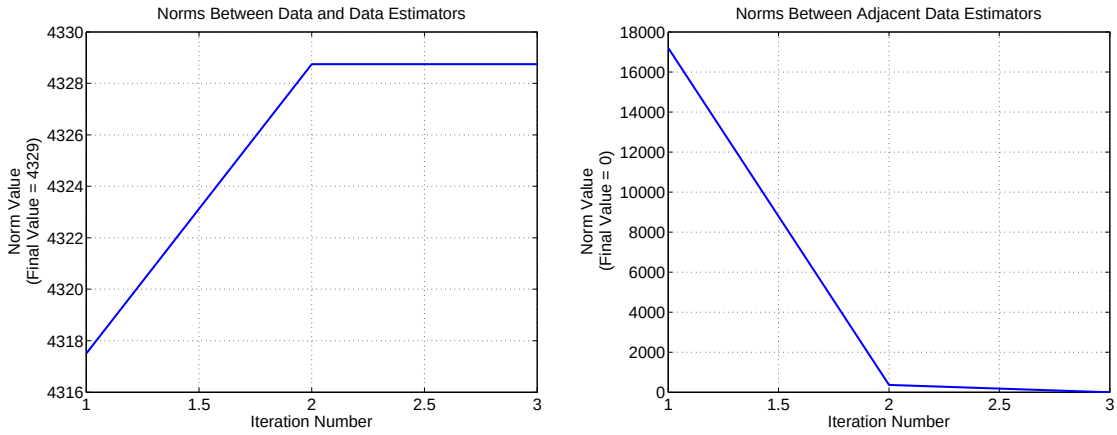
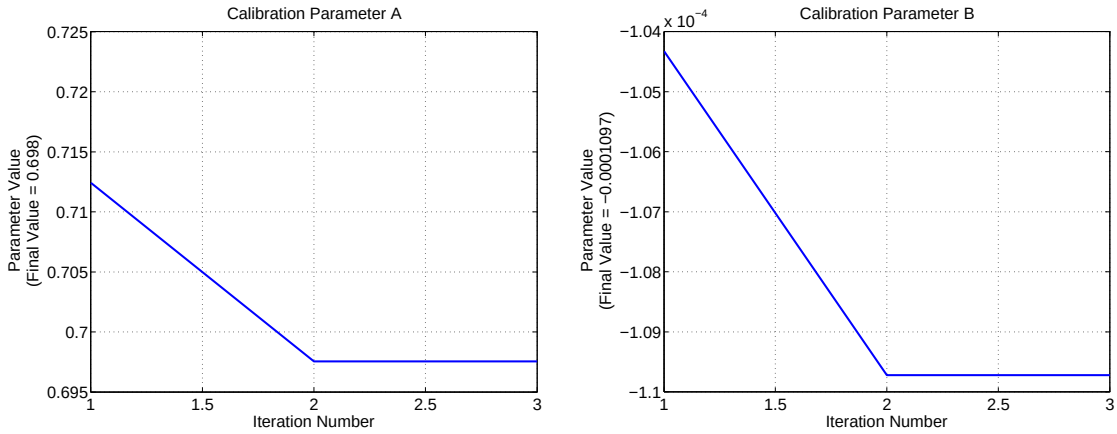


Figure 17: Scene used to test the joint estimation of eigentank expansion coefficients and FLIR calibration parameters of a single T-62 tank.



(a) (b)

Figure 18: Figures (a) and (b) show how the T-62 estimated calibrated data changes with each iteration of the Levenberg-Marquardt algorithm. Figure (a) shows the norm between the estimated data values and the ground truth for each facet of the T-62 tank. Figure (b) shows the norm of the first difference between estimated data values.



(a) (b)

Figure 19: Figures (a) and (b) show how the T-62 calibration terms change with each iteration of the Levenberg-Marquardt algorithm. Figure (a) shows the absolute difference between the estimated and ground-truth calibration term a, while Figure (b) shows the absolute difference between the estimate and ground-truth calibration term b.

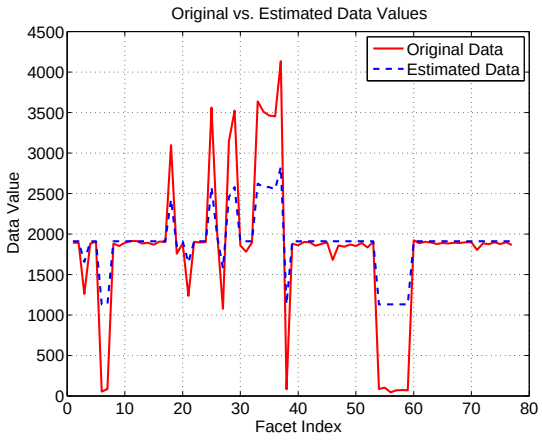


Figure 20: Comparison of the original thermal data and the thermal estimates at each facet index of an T-62 tank, obtained via the Levenberg-Marquardt algorithm. Facet indices are not listed in any particular order.

4.6.3 Case 3: Both an M-60 and a T-62

Figure 21 shows a synthetic scene is generated with a T-62 tank and a M-60 tank both placed on a flat ground plane. Once again, all simulation parameters are the same as those specified in Sections 4.6.1 and 4.6.2. We use both a T-62 and M-60 in this test to explore the effectiveness of the algorithm when operating with multiple known target classes. As shown in Figures 22 to 24, the addition of the second tank does not appear to degrade the estimation performance, but the algorithm did not perform very well to begin with. Since the Levenberg-Marquardt algorithm contains empirically determined parameter values, poor performance might be directly related to the values used for these tuning parameters. Instead of further investigating the adjustment of tuning parameters to improve the performance, we chose to an alternate path and pursued a different kind of algorithm, as discussed in the next section.

4.7 Simplified Linear Solutions to the Nonlinear System

As shown in the previous section, jointly estimating both the expansion coefficients and calibration terms using standard techniques for systems of nonlinear equations can be problematic. If we look at the set of equations (37) and (48) separately, we

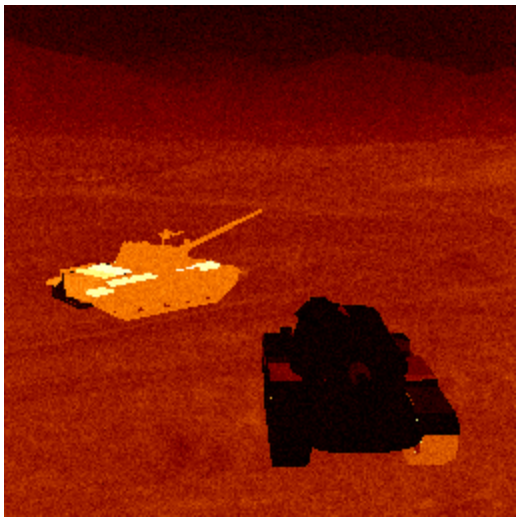
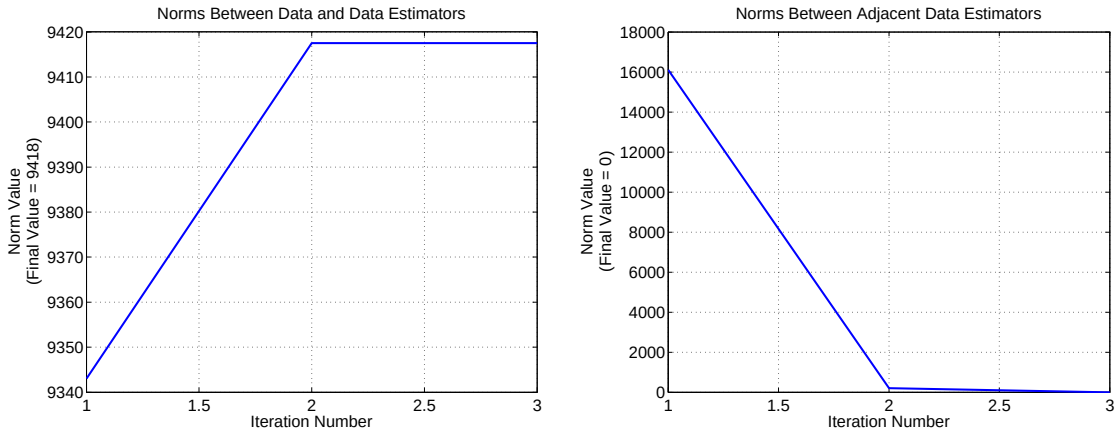
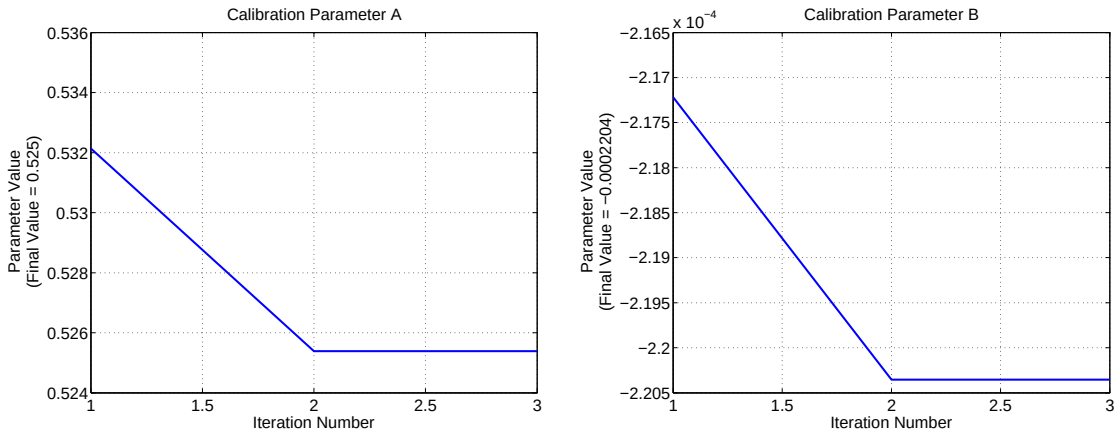


Figure 21: Scene used to test the joint estimation of eigentank expansion coefficients and FLIR calibration parameters of both an M-60 tank and a T-62 tank.



(a) (b)

Figure 22: Figures (a) and (b) show how the estimated calibrated data for a scene containing both an M-60 and a T-62 changes with each iteration of the Levenberg-Marquardt algorithm. Figure (a) shows the norm between the estimated data values and the ground truth for each facet of both tanks. Figure (b) shows the norm of the first difference between estimated data values.



(a) (b)

Figure 23: Figures (a) and (b) show how the calibration terms for the combined M-60 and T-62 test scene change with each iteration of the Levenberg-Marquardt algorithm. Figure (a) shows the absolute difference between the estimated and ground-truth calibration term a, while Figure (b) shows the absolute difference between the estimate and ground-truth calibration term b.

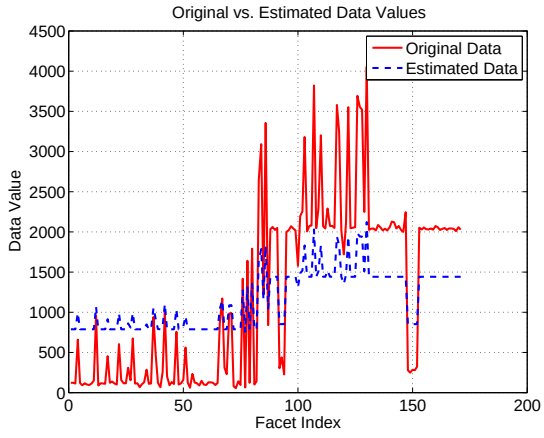


Figure 24: Comparison of the original thermal data and the thermal estimates at each facet index of both an M-60 tank and a T-62 tank, obtained via the Levenberg-Marquardt algorithm. Facet indices are not listed in any particular order.

can see a linear structure that we can exploit. It is immediately apparent that the conditions specified in (48) can be represented in the form of a matrix to solve for a and b, where the $\alpha_{j,t}$ terms are help constant. These equations represent the least squares solution to the problem of determining the affine parameters a and b that best fit the derived thermal intensities $\lambda_{i,t}$ from the data $d(k)$.

Conversely, if we keep the a and b terms constant and optimize over the $\alpha_{j,t}$ terms, The equations in (37) represent T sets of linear equations, one set for each target, conveniently expressed as the following matrix:

$$\left(\begin{bmatrix} \sum_i N_{i,t}\Phi_{i1,t}^2 & \cdots & \sum_i N_{i,t}\Phi_{iJ,t}\Phi_{i1,t} \\ \vdots & & \vdots \\ \sum_i N_{i,t}\Phi_{i1,t}\Phi_{iJ,t} & \cdots & \sum_i N_{i,t}\Phi_{JJ,t}^2 \end{bmatrix} + diag\left(\begin{bmatrix} \frac{\sigma^2}{a^2\gamma_{1,t}} \\ \vdots \\ \frac{\sigma^2}{a^2\gamma_{J,t}} \end{bmatrix}\right)\right)\begin{bmatrix} \alpha_{1,t} \\ \vdots \\ \alpha_{J,t} \end{bmatrix}$$
$$= \begin{bmatrix} \sum_i \Phi_{i1,t}(\frac{D_{i,t}-bN_{i,t}}{a} - N_{i,t}m_{i,t}) \\ \vdots \\ \sum_i \Phi_{iJ,t}(\frac{D_{i,t}-bN_{i,t}}{a} - N_{i,t}m_{i,t}) \end{bmatrix}. \quad (61)$$

We now have two sets of linear equations: one set that maximizes the logposterior H with respect to the α terms when the a and b terms are held constant, and another set that maximizes the same logposterior with respect to the a and b terms when the α terms are held constant. Written together, these sets of equations represent a system of nonlinear equations. Many techniques exist to find solutions to such a system, but as seen in Section 4.6 on the Levenberg-Marquardt algorithm, such techniques can be difficult to implement for various reasons. Since this linear system separates so nicely into two systems of linear equations, it is reasonable to conjecture that there may be an iterative technique to solve for the expansion coefficients and calibration terms in separate stages. If we can determine an appropriate initial value, we can use the algorithm shown in Algorithm 3 to iteratively estimate the expansion coefficients and calibration terms.

The technique described in this section is sensitive to the choice of parameter

Algorithm 3 Algorithm to compute the thermal expansion coefficients α_j and thermal calibration coefficients a and b using the simplified sets of linear equations.

Compute $N_{i,t}$ and $D_{i,t}\ \forall\ i,t$
$\alpha_{j,t}^{(n)} \leftarrow 0\ \forall\ j,t$
$a^{(n)} \leftarrow 1$
$b^{(n)} \leftarrow 0$
$\Lambda_{i,t}^{(n)} \leftarrow 0$
repeat
    $a^{(n-1)} \leftarrow a^{(n)}$
    $b^{(n-1)} \leftarrow b^{(n)}$
    $\Lambda_{i,t}^{(n-1)} \leftarrow \Lambda_{i,t}^{(n)}$
    $\alpha_{j,t}^{(n-1)} \leftarrow \alpha_{j,t}^{(n)}\ \forall\ j,t$
    Solve for the $\alpha_{j,t}^{(n)}$ coefficients using $a^{(n-1)}$ and $b^{(n-1)}\ \forall\ j,t$
    Compute $\lambda_{i,t} = \sum_j \alpha_{j,t}^{(n)} \Phi_{ij,t} + m_{i,t}\ \forall\ i,t$
    Solve for $a^{(n)}$ and $b^{(n)}$ using $\lambda_{i,t}$, $N_{i,t}$, and $D_{i,t}\ \forall\ i,t$
    Compute $\Lambda_{i,t}^{(n)} = a^{(n)}\lambda_{i,t} + b^{(n)}$
until $|\Lambda_{i,t}^{(n)} - \Lambda_{i,t}^{(n-1)}| < \epsilon$

values. The iteration may easily converge to an incorrect solution by finding one of the local maxima present in the posterior distribution. Most likely, we can only guarantee convergence if the distribution is unimodal, i.e., having a single local maximum. In that case, every iterative step should result in parameter estimates that increase the posterior value.

The simplified linear approach has a number of advantages over the Levenberg-Marquardt algorithm. First, it contains no parameters that need to be tuned. Though the iteration parameters used in the Levenberg-Marquardt algorithm can be adjusted to work in many situations, the algorithm presented in this section does not require spending time to find those parameters. Second, the size of the Jacobian matrix used in the Levenberg-Marquardt algorithm increases with each additional target by the number of expansion coefficients $\alpha_{J_T,T}$ for that target. Large matrices can become ill-conditioned, or otherwise difficult to implement because of memory and computation requirements. In the simplified linear solution, each additional target means the algorithm must compute another set of expansion coefficients. But the

system of equations used to compute expansion coefficients is relatively simple and maintains a straightforward parallel implementation.

4.8 Simulations of the Simplified Linear Solutions

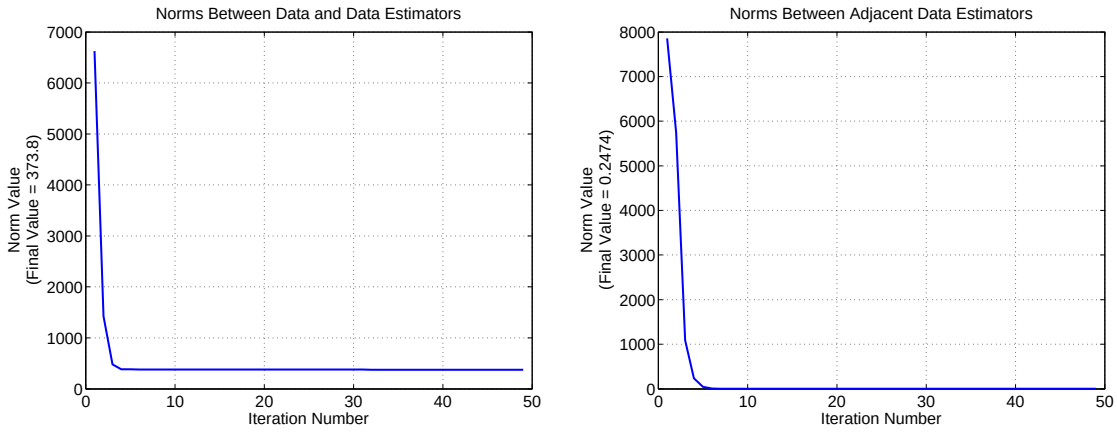
This section presents simulations of Algorithm 3 under the same conditions presented in Section 4.6, using the same three test cases.

4.8.1 Case 1: A Single M-60

A synthetic scene with a single M-60 tank is placed on a flat ground plane under the same conditions as specified in Section 4.6.1.

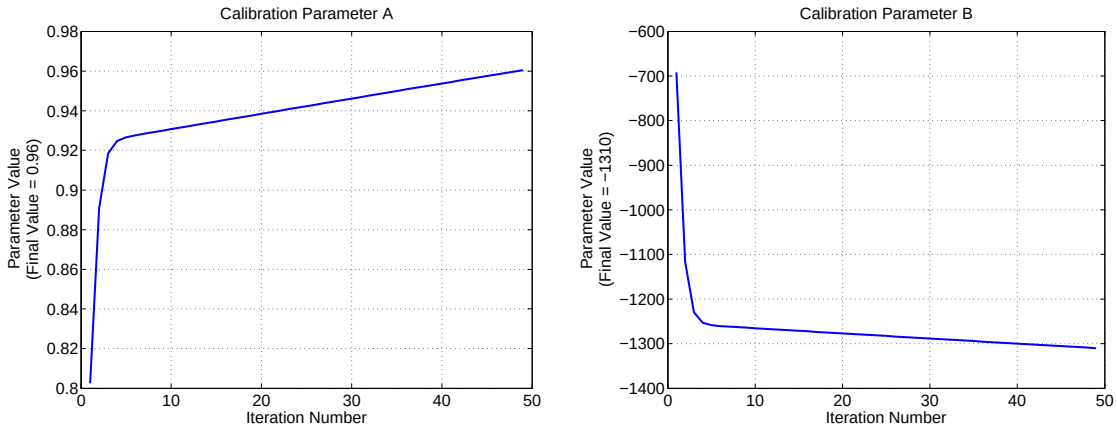
Figure 25 shows the convergence of the expansion coefficients $\alpha_{j,t}$ and calibration terms a and b using the simplified linear solution. When compared to the plot from Levenberg-Marquardt approach scenario shown in Figure 14, we can see that the norms between the original data facet values $D_{i,t}$ and the estimated data value $\Lambda_{i,t}$ decrease with each iteration step using this approach. Interestingly, the calibration terms do not appear to converge to a fixed value, as show in Figure 26. This appears to indicate that while a single set of calibration terms and expansion coefficients were used to generate the original test data, the a, b, and possibly $\alpha_{j,t}$ solutions to the simplified linear system may not be unique. The result is that the iteration simply attempts to minimize the norm between adjacent data estimators.

Figure 27 compares the final target intensity estimates with those from the original data scene. It is clear that the results of the simplified linear algorithm are much better than those obtained from the Levenberg-Marquardt algorithm. We can see that the estimated thermal values at each facet index are much closer to the ground-truth values.



(a) (b)

Figure 25: Figures (a) and (b) show how the M-60 estimated calibrated data changes with each iteration of the simplified linear algorithm. Figure (a) shows the norm between the estimated data values and the ground truth for each facet of the M-60 tank. Figure (b) shows the norm of the first difference between estimated data values.



(a) (b)

Figure 26: Figures (a) and (b) show how the M-60 calibration terms change with each iteration of the simplified linear algorithm. Figure (a) shows the absolute difference between the estimated and ground-truth calibration term a, while Figure (b) shows the absolute difference between the estimate and ground-truth calibration term b.

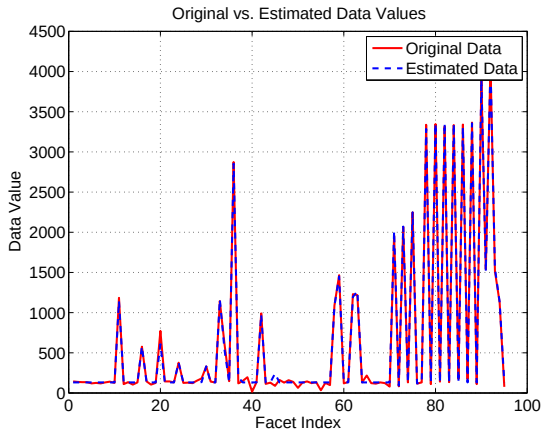


Figure 27: Comparison of the original thermal data and the thermal estimates at each facet index of an M-60 tank, obtained via the simplified linear algorithm. Facet indices are not listed in any particular order.

4.8.2 Case 2: A Single T-62

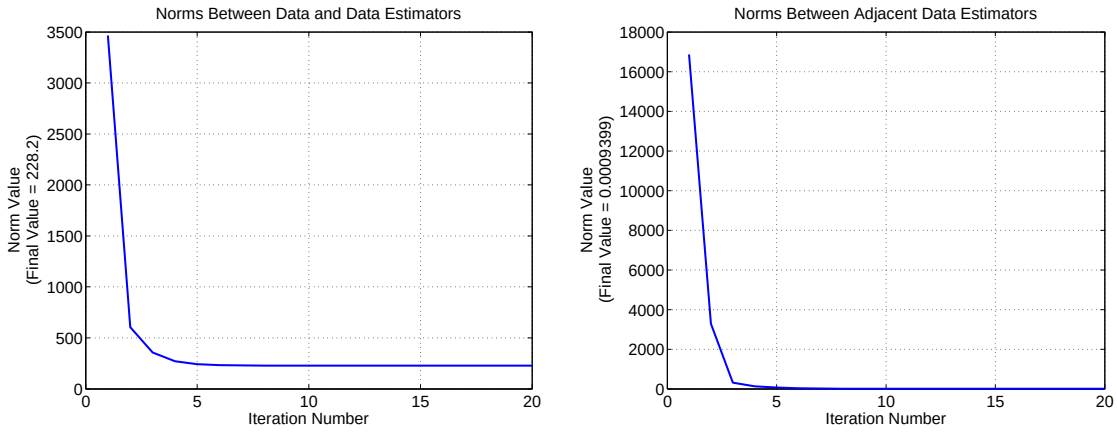
A synthetic scene with a single T-62 tank is placed on a flat ground plane in the same position as the M-60 from the previous section. The same simulation parameters used in Section 4.6.2 are also used here to define the scene.

As seen in Figure 29, we once again see that the a and b calibration terms to do not converge to the values generate the original dataset. Unlike the M-60 case, these do appear to converge to a fixed value.

Figure 30 shows the comparison between the estimated thermal values and the original data, at each facet index. We once again see that the simplified linear solution performs significantly better than the Levenberg-Marquardt solution on the T-62 data.

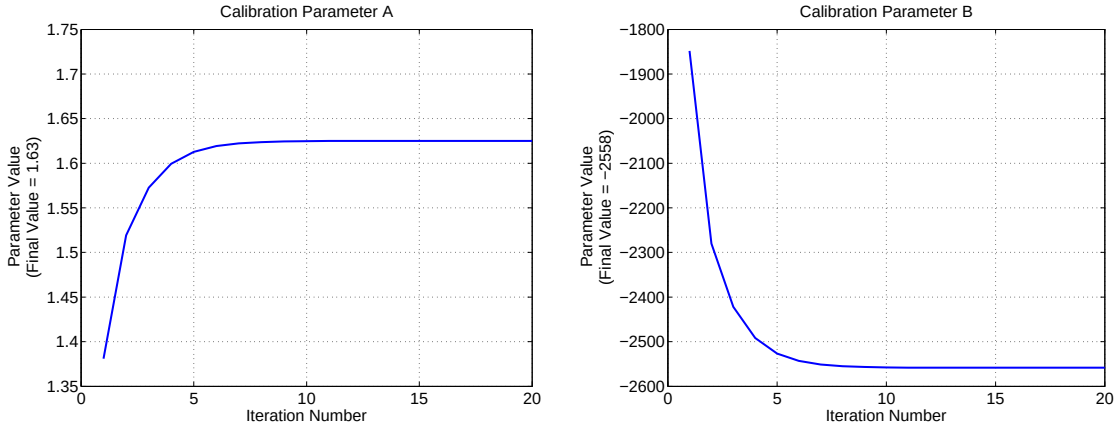
4.8.3 Case 3: Both an M-60 and a T-62

The same scene containing as described in Section 4.6.3, containing both an M-60 and a T-62, is also used here. In Figures 31, 32, and 33, we see the same trends as in the previous cases where the simplified linear solution continues to outperform the Levenberg-Marquardt solution. The addition of the second target does not hinder



(a) (b)

Figure 28: Figures (a) and (b) show how the T-62 estimated calibrated data changes with each iteration of the simplified linear algorithm. Figure (a) shows the norm between the estimated data values and the ground truth for each facet of the T-62 tank. Figure (b) shows the norm of the first difference between estimated data values.



(a) (b)

Figure 29: Figures (a) and (b) show how the T-62 calibration terms change with each iteration of the simplified linear algorithm. Figure (a) shows the absolute difference between the estimated and ground-truth calibration term a, while Figure (b) shows the absolute difference between the estimate and ground-truth calibration term b.

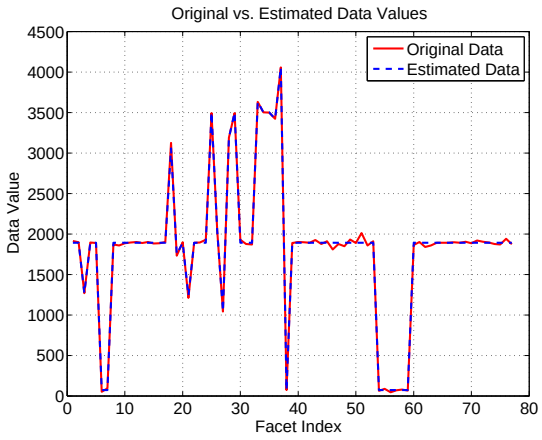
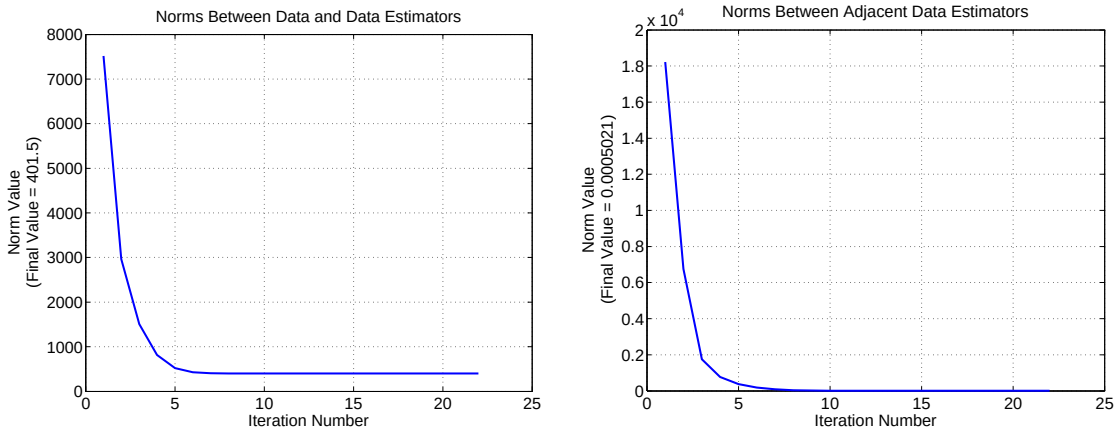


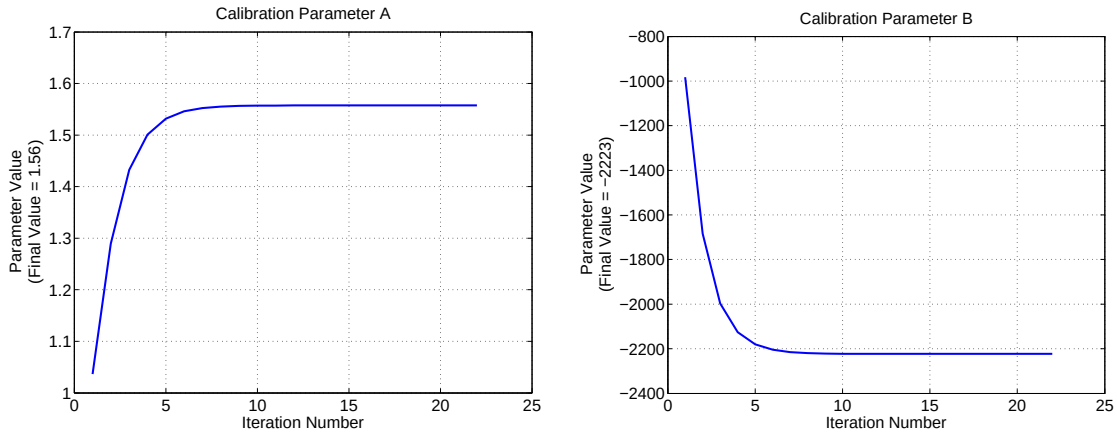
Figure 30: Comparison of the original thermal data and the thermal estimates at each facet index of a T-62 tank, obtained via the simplified linear algorithm. Facet indices are not listed in any particular order

the algorithm's performance.



(a) (b)

Figure 31: Figures (a) and (b) show how the combined M-60 and T-62 estimated calibrated data changes with each iteration of the simplified linear algorithm. Figure (a) shows the norm between the estimated data values and the ground truth for each facet of the M-60 tank. Figure (b) shows the norm of the first difference between estimated data values.



(a) (b)

Figure 32: Figures (a) and (b) show how the combined M-60 and T-62 calibration terms change with each iteration of the simplified linear algorithm. Figure (a) shows the absolute difference between the estimated and ground-truth calibration term a, while Figure (b) shows the absolute difference between the estimate and ground-truth calibration term b.

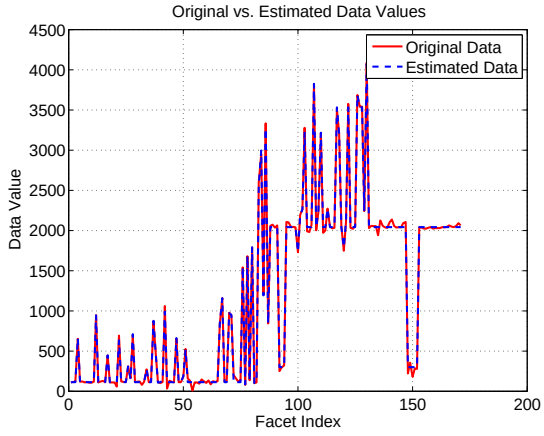


Figure 33: Comparison of the original thermal data and the thermal estimates at each facet index of both an M-60 tank and a T-62 tank, obtained via the simplified linear algorithm. Facet indices are not listed in any particular order

4.9 Conclusion

This chapter discussed the problem of FLIR sensor calibration with respect to the estimation of radiance profile of targets. We have shown two methods for jointly estimating the thermal expansion coefficients and calibration parameters using the scene data, eigentank expansion information, and Gaussian noise statistics. The Levenberg-Marquardt algorithm is a standard method for solving systems of nonlinear equations, but its application to this problem showed less than promising results. Fortunately, the form of the system of nonlinear equations relating the terms of interest can be reduced two simplified linear equations that may be solved iteratively. Additionally, the jump-diffusion with the thermal inference implementation described in Section 2.2.3 can be modified to include inference of the a and b calibration parameters, thus allowing the jump-diffusion ATR algorithm to apply to uncalibrated FLIR imagery. This implementation is left as an activity for future work.

Two additional considerations should be taken when investigating this problem in the future. The first is the inclusion of constraints on the calibration terms and expansion coefficients. The discussion in this chapter treated the parameter estimation problem as an unconstrained optimization problem, allowing for the possibility that some estimated calibration parameters and expansion coefficients can lead to thermal profiles that defy the physics of the thermal imaging device (e.g., negative thermal intensities). The other consideration is the possibility of an Expectation-Maximization (EM) Algorithm [11] for solving this problem. EM algorithms are often used to solve problems of this form, and are popular because of their theoretical elegance. We believe the the simplified linear solution discussed previously has many similarities to the EM algorithm and may be comparable in terms of its convergence properties. Should an EM derivation be found, it would be appropriate to compare it to the techniques discussed in this chapter (it may even be the case that our algorithm can be framed as an EM algorithm).

The Bayesian framework we have established for pattern-theoretic ATR allows for naturally expanding the recognition capacity of an algorithm by including more terms in the the sensor-likelihood function. In this chapter, the addition of calibration terms allows us to relax assumptions on the thermal state of targets of interest. The unfortunate consequence of this simple action is that the complexity of applying the sensor likelihood is also increased. We have shown that certain approaches to parameter estimation problems can lead to a large number of variables that must be considered, as is the case of applying the Levenberg-Marquardt solution to the calibration problem. We were fortunate to find a relatively simple solution to the system of nonlinear equations created through applying the calibration terms, but there is no guarantee that considering additional sensor characteristics (for instance, extending the affine calibration mapping to a quadratic mapping) will lead to a similarly convenient result.

CHAPTER V

ACCURACY BOUNDS ON POSE PARAMETER ESTIMATION

5.1 Introduction

This chapter is devoted to the development of approximate Cramér-Rao lower bounds on the accuracy of estimating the pose (position and orientation) of rigid targets via range imagery collected by a coherent-detection laser radar. The bounds are based on the likelihood model presented in Section 1.3.2, incorporating both the sensor statistics as well as the perspective projection imaging process. We examine how the bounds change as a function of range, resolution, and target orientation, particularly exploring the coupling between various pose parameter estimates. We also compare our Cramér-Rao bounds to the performance of a simple parameter estimation algorithm calculated via Monte Carlo simulation.

The development of ATR algorithms necessitates methods to compare algorithm effectiveness. Without a measure of performance, users of target recognition systems have no way of identifying how their algorithms compare with others, or determining if the sensor in use needs to be improved to achieve better target recognition results. It is also useful to know the fundamental limit on the ability to estimate a target's parameters of interest given a sensor's operating conditions.

5.2 Background

A number of studies have used information-theoretic bounds to generate performance and accuracy predictions for different types of sensors. Koksal et al. [36] used performance bounds to determine pose estimation accuracy from FLIR and LADAR. In

these experiments, Cramér-Rao bounds were compared to Hilbert-Schmidt bounds on the mean squared error of estimating orientation as a function of noise [22]. Jain et al. used a number of information-theoretic tools to determine bounds on target detection performance from images with noise and clutter [33]. The bounds noted in that study included Kullback-Leibler and Chernoff distances. It considered both cases without nuisance parameters and cases with the nuisance parameter of orientation. This dissertation focuses on standard deviation bounds for pose estimation by applying the Cramér-Rao lower bound. Although most ATR algorithms consider pose parameters to be nuisance parameters, we consider them directly because of their importance in aimpoint selection or other scenarios where it is necessary to identify specific points on a target. We investigate how the bounds change as we vary the distance, position, and orientation of the targets. A similar study using CRLBs was performed by Gerwe et al. [16, 17] to determine bounds on orientation when viewing satellites from ground-based optical sensors. Other works have used information-theoretic concepts to analyze a sensor for ATR by examining the information content of imagery acquired by the sensor [7].

5.3 Derivation of the Cramér-Rao Bound

The matrix Cramér-Rao bound for a vector of unbiased estimators is defined as the inverse of the Fisher information matrix (FIM) for that vector, where each element of the FIM is defined as

$$F_{ij}(\boldsymbol{\Theta}) = E\left[\left(\frac{\partial}{\partial \Theta_i}\log p(D;\boldsymbol{\Theta})\right)\left(\frac{\partial}{\partial \Theta_j}\log p(D;\boldsymbol{\Theta})\right)\right], \tag{62}$$

where $p(d;\boldsymbol{\Theta})$ is the data loglikelihood, which is a function of the target parameters $\boldsymbol{\Theta} = [x, y, \theta]^T$. We use a capital D in (62) to indicate that we are using a random variable for d. The coordinates x and y denote the target location on the ground plane, which is assumed to be flat, and θ is the orientation angle with respect to the axis that points out of the ground plane. We start with the laser radar sensor

likelihood equation from Section 1.3.2, repeated here for convenience:

$$L_{LR}(\mathbf{d}|\boldsymbol{\mu}) = \sum_n -\frac{1}{2}\log\left\{2\pi\sigma^2(n)\right\} - \frac{[d(n)-\mu(n)]^2}{2\sigma^2(n)}. \tag{63}$$

Let render($\boldsymbol{\Theta}$) denote the operation of simulating an uncorrupted LADAR range image of a scene with parameter vector $\boldsymbol{\Theta}$, as seen through the effects of perspective projection and obscuration. In our framework, $\mu(n)$ is a function of $\boldsymbol{\Theta}$, which we emphasize by switching to the notation $\mu(n;\boldsymbol{\Theta}) = render(\boldsymbol{\Theta})$. To derive the Cramér-Rao lower bound, we begin by determining the derivatives of the LADAR loglikelihood function with respect to the parameters in vector $\boldsymbol{\Theta}$. To make the calculation simpler, we compute $\sigma^2(n)$ for a given $\boldsymbol{\Theta}$ and approximate it as *constant* with respect to small changes in $\boldsymbol{\Theta}$. Using this approximation, the first term in (63) may be dropped and the derivative can be computed as

$$\frac{\partial}{\partial\Theta_i}L_{LR}(\mathbf{D};\boldsymbol{\mu}) = \sum_n \frac{-[D(n)-\mu(n;\boldsymbol{\Theta})]\frac{\partial}{\partial\Theta_i}\mu(n;\boldsymbol{\Theta})}{\sigma^2(n)}. \tag{64}$$

To simplify the expressions that follow, define

$$G(n;\boldsymbol{\Theta}) = \frac{D(n)-\mu(n;\boldsymbol{\Theta})}{\sigma^2(n)}. \tag{65}$$

Our sensor likelihood model treats the data scene $D(n)$ as a Gaussian random variable with a mean given by the uncorrupted range $\mu(n;\boldsymbol{\Theta})$ and variance $\sigma^2(n)$ for pixel n. Therefore, $E[D(n)] = \mu(n;\boldsymbol{\Theta})$. Using this relationship, the FIM becomes

$$F_{ij}(\boldsymbol{\Theta}) = E\left[\sum_n\left(-G(n;\boldsymbol{\Theta})\frac{\partial}{\partial\Theta_i}\mu(n;\boldsymbol{\Theta})\right)\sum_{\tilde{n}}\left(-G(\tilde{n};\boldsymbol{\Theta})\frac{\partial}{\partial\Theta_j}\mu(\tilde{n};\boldsymbol{\Theta})\right)\right] \tag{66}$$

$$= E\left[\sum_n\sum_{\tilde{n}} G(n;\boldsymbol{\Theta})G(\tilde{n};\boldsymbol{\Theta})\frac{\partial}{\partial\Theta_i}\mu(n;\boldsymbol{\Theta})\frac{\partial}{\partial\Theta_j}\mu(\tilde{n};\boldsymbol{\Theta})\right] \tag{67}$$

$$= \sum_n\sum_{\tilde{n}} E\left[G(n;\boldsymbol{\Theta})G(\tilde{n};\boldsymbol{\Theta})\right]\frac{\partial}{\partial\Theta_i}\mu(n;\boldsymbol{\Theta})\frac{\partial}{\partial\Theta_j}\mu(\tilde{n};\boldsymbol{\Theta}). \tag{68}$$

The expectation in this last equation appears in the form of a covariance between the random variables $D(n)$ and $D(\tilde{n})$. We assume that pixels in the LADAR image

are conditionally independent, so the covariance is zero when $n \neq \tilde{n}$. Therefore, each element in the FIM can be reduced to

$$F_{ij}(\mathbf{\Theta}) = \sum_{n} \frac{\frac{\partial}{\partial \Theta_i}\mu(n;\mathbf{\Theta})\frac{\partial}{\partial \Theta_j}\mu(n;\mathbf{\Theta})}{\sigma^2(n)}. \tag{69}$$

Taking the inverse of this matrix leaves us with a *lower bound* on the covariance matrix for the parameters of interest. By saying that the covariance matrix is *bounded* by the inverse FIM, we mean that $\mathrm{Cov}(\mathbf{\Theta}) \geq \mathbf{F}(\mathbf{\Theta})^{-1}$ (i.e., the matrix $\mathrm{Cov}(\mathbf{\Theta}) - \mathbf{F}(\mathbf{\Theta})^{-1}$ is nonnegative definite for unbiased estimators[1]). This implies that the diagonal elements of the covariance matrix are greater than or equal to the diagonal elements of the inverse FIM. The diagonal elements of the inverse FIM are the Cramér-Rao lower bounds for the corresponding variances in the covariance matrix.

Cramér-Rao lower bounds are, theoretically, only defined for flat Euclidean spaces. Gerwe et al. [16] note that, in practice, CRLBs can be applied to curved spaces if the bound is relatively small compared to the possible range of values in the space. Since we are concerned with orientation angles, we only consider cases where the angle bound is much less than 180 degrees. Note that the maximum error one can have in orientation is 180 degrees.

One benefit of using Cramér-Rao lower bounds in this manner is that we can adapt them for any type of sensor, assuming that the sensor likelihood function between the uncorrupted data values and the measured data values is known. Sensor fusion is naturally incorporated by simply adding the loglikelihoods of the individual sensors.

5.4 Cramér-Rao Bound Implementation Details

We used OpenGL and faceted CAD models to simulate laser radar imagery in the same manner as discussed in Section 2.1.2. To compute the derivatives necessary for

[1]There are versions of the Cramér-Rao bound that incorporate bias effects; however, these generally involve analytical expressions of derivatives of the bias, which are not available, so they are rarely used. Even in situations with biased estimators, the "unbiased" version of the Cram'er-Rao bound often illustrates useful trends.

determining the CRLBs, a finite difference approximation is employed. Remembering that $\boldsymbol{\mu}$ is a complicated, nonlinear function of the parameter vector $\boldsymbol{\Theta}$ obtained from render($\boldsymbol{\Theta}$), we apply a finite difference approximation similar to (11):

$$\frac{\partial \mu}{\partial \Theta_i} = \frac{\partial \text{render}(\boldsymbol{\Theta})}{\partial \Theta_i} \approx \frac{\text{render}(\ldots, \Theta_i + \delta, \ldots) - \text{render}(\ldots, \Theta_i - \delta, \ldots)}{2\delta}, \tag{70}$$

where δ is some small deviation of the parameter Θ_i, which is a component of $\boldsymbol{\Theta}$, and the ellipses indicate that the remaining parameters are held fixed. The derivative provides us with a representation of how the image changes with small changes of the pose parameters.

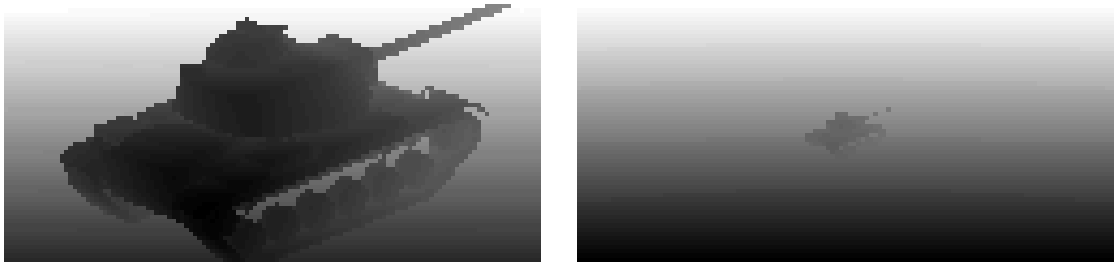
5.5 CRLB Experiments

For the experiments that follow, we assume the laser radar parameters found in Table 1, which were taken from the coherent-detection forward-looking laser radar discussed by Bounds [3]. We assume the laser radar points toward a single M60 tank with a vertical field of view of 12 milliradians. The tank parameter space $\boldsymbol{\Theta}$ has variables x, y, and θ, where x and y are the crossrange and downrange coordinates on the ground, while θ is the angular orientation in degrees around the axis pointing out of the ground plane. All parameters are referenced to some known initial position and orientation. Since we assume the target is sitting on the ground plane, the z-coordinates of our targets are always zero. (Future work could explore relaxing this assumption, which might involve including additional pose parameters.) The size of the image obtained from the laser radar is assumed to be 125 x 60 pixels. Sample images of the tank viewed from an elevation angle of 15 degrees with the maximum and minimum ranges studied, 375m and 2 km, respectively, are shown in Figure 34.

Our first experiment estimated CRLBs for the θ orientation angle, x-crossrange position, and y-downrange position for a target of interest while varying the distance from the laser radar sensor to the target. Each data set was assumed to have a single target sitting on a ground plane with a fixed x position centered on the line

Table 1: Parameters for the LADAR statistical model. The values are based on the system parameters for a forward-looking short-pulse coherent-detection system.

Parameter	Value
Optical Efficiency, ϵ_{opt}	0.5
Heterodyne Efficiency, ϵ_{het}	0.5
Detector Quantum Efficiency, η	0.25
Receiver Aperture Dimension, D_R	13 cm
Atmospheric Extinction Coefficient, α	0.5 dB/km
Average Transmitted Power, P_s	5 W
IF Filter Bandwidth, B	80 MHz
Photon Energy, $h\nu$	1.87×10^{-20} J
Range Resolution, R_{res}	0.6 m
Target Reflectivity, ρ	0.25



(a) (b)

Figure 34: Views of the M60 tank at (a) a position close to the laser radar sensor and (b) a position far from the laser radar sensor. These images are shown to illustrate the coverage over the sensor's field of view and do not include the noise from the LADAR statistical model defined in Table 1.

of sight of the LADAR sensor. When computing the Cramér-Rao lower bounds, we considered cases where the parameters are coupled (pairwise parameter estimates have nonzero covariance when $\Theta_i \neq \Theta_j$) as well as cases where the parameters are decoupled (pairwise parameter estimates have zero covariance when $\Theta_i \neq \Theta_j$). The decoupled parameter cases are artificial in that they assume the other parameters are known precisely; comparing the decoupled case against the coupled case indicates how much information is lost from having to co-estimate the other parameters. To present bounds for the x and y parameters, we compute the CRLB for a set of equally spaced angles through a full 360 degree rotation and present the average of the CRLBs. One could also present similar results for specific rotations of interest.

In the second experiment, we fix the laser radar sensor and the target at a single location. Bounds on the pose parameters are computed at each orientation angle at fixed intervals between zero and 360 degrees to determine how our estimation ability changes as we view the target from different angles. This second experiment only considers the location closest to the LADAR.

The next two subsections present charts showing how the bounds on the standard deviation of pose parameters change with orientation angle and distance from target. We suggest that the absolute values of these bounds are not as significant as the relative trends. The LADAR model that we employ does not account for all possible sources of noise and image corruption, so the bounds are much lower than one might expect an algorithm to perform in practice.

5.5.1 CRLB vs. Range from Target Results

The Cramér-Rao lower bounds on pose estimation as a function of range from the target of interest behave as one would expect: bounds tend to increase as range increases. This can be attributed to the decrease in target information provided by the sensor (i.e., fewer pixels on target), because as the sensor moves further away the

target becomes smaller within the limits of the field of view. The equations for the LADAR statistics also tell us that the local range accuracy worsens with range, thus increasing the CRLB even further. This result is consistent for all parameters, as can be seen in Figure 35. Figure 35(d) shows the number of target pixels present in the image at the given ranges from the LADAR sensor.

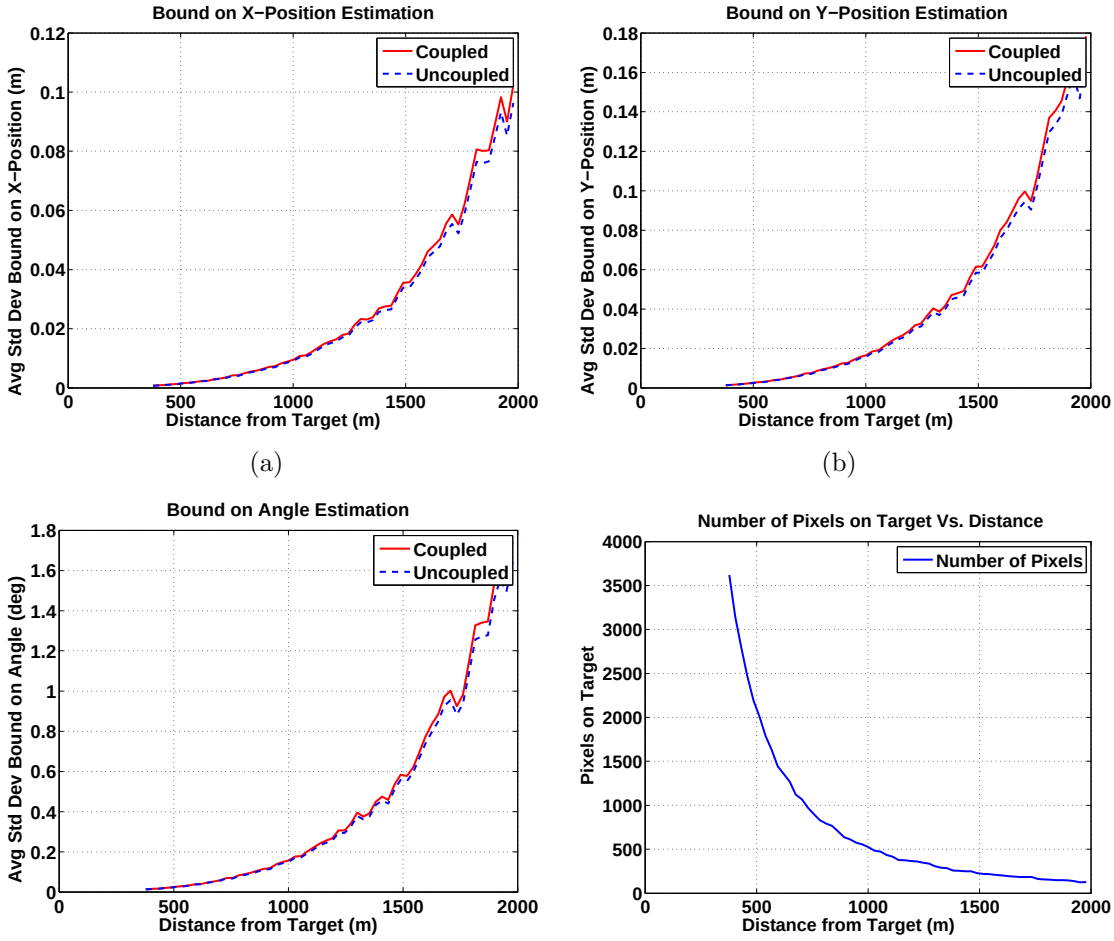


Figure 35: Cramér-Rao lower bounds on estimating (a) x-position, (b) y-position, and (c) orientation angle for a single M60 tank with respect to distance between the LADAR sensor and the tank, and (d) pixels on target versus distance when the M60 is at an orientation angle of 0 degrees (pointing to the right).

For larger range values, the curve is not monotonically increasing; small dips appear sporadically along the curve. This may be due to a number of factors, including limitations of the rendering process. Within a small range interval, it is possible

that different features of the object being viewed will appear on the final image, and the pixels containing these features may contain different amounts of information. A more theoretical analysis of spatial resolution and sampling may be necessary to determine the exact cause; some explorations along these lines will be illustrated in Section 5.5.3. These non-monotonic qualities of the CRLB curve may also be related to derivative approximations involved in computing the Fisher information matrix. It is clear that the bounds may change when adjusting the derivative stepsizes, but the overall effect of such changes is complicated, as Section 5.5.4 will illustrate. Some of these issues also arise in work by Gerwe et al. [17]

Chapter 3 discussed a pixel-based approached to pose-parameter refinement that replaces the approximation of the Langevin SDE used in the diffusion process. This approach allowed us to by bypass the finite difference derivative calculation and the determination of arbitrary stepsizes for that calculation. Future work may focus on applying the pixel-based approach to determining a better set of location-specific stepsizes to use when computing the finite difference approximation used in CRLB calculation.

5.5.2 CRLB vs. Target Orientation Results

Figure 36 shows how the CRLB changes with the target's orientation angle. When estimating the bounds on the x-position and orientation angle, we see that the CRLBs vary with changes to these parameters, although the variation is difficult to intuitively explain. When estimating the y-position, we see noticeable peaks around 90 degrees and 270 degrees. In those instances, the M60 is either pointing toward the LADAR sensor or away from it. This may suggest that the decrease in visible tank surface area increases the difficulty in estimating the target's ground distance from the sensor.

It is interesting to note how the bound changes if parameters are coupled or decoupled. We see, as expected, that the bounds are higher when the parameters

must be jointly estimated. In some cases, the bounds are extremely close, as seen with the bounds on the y-position and orientation angle estimation in Figures 36(b) and 36(c), respectively. In the case of estimating the x-position in Figure 36(a), there are many noticeable gains if the parameters can be individually estimated. Of course, this will rarely be the case in practice.

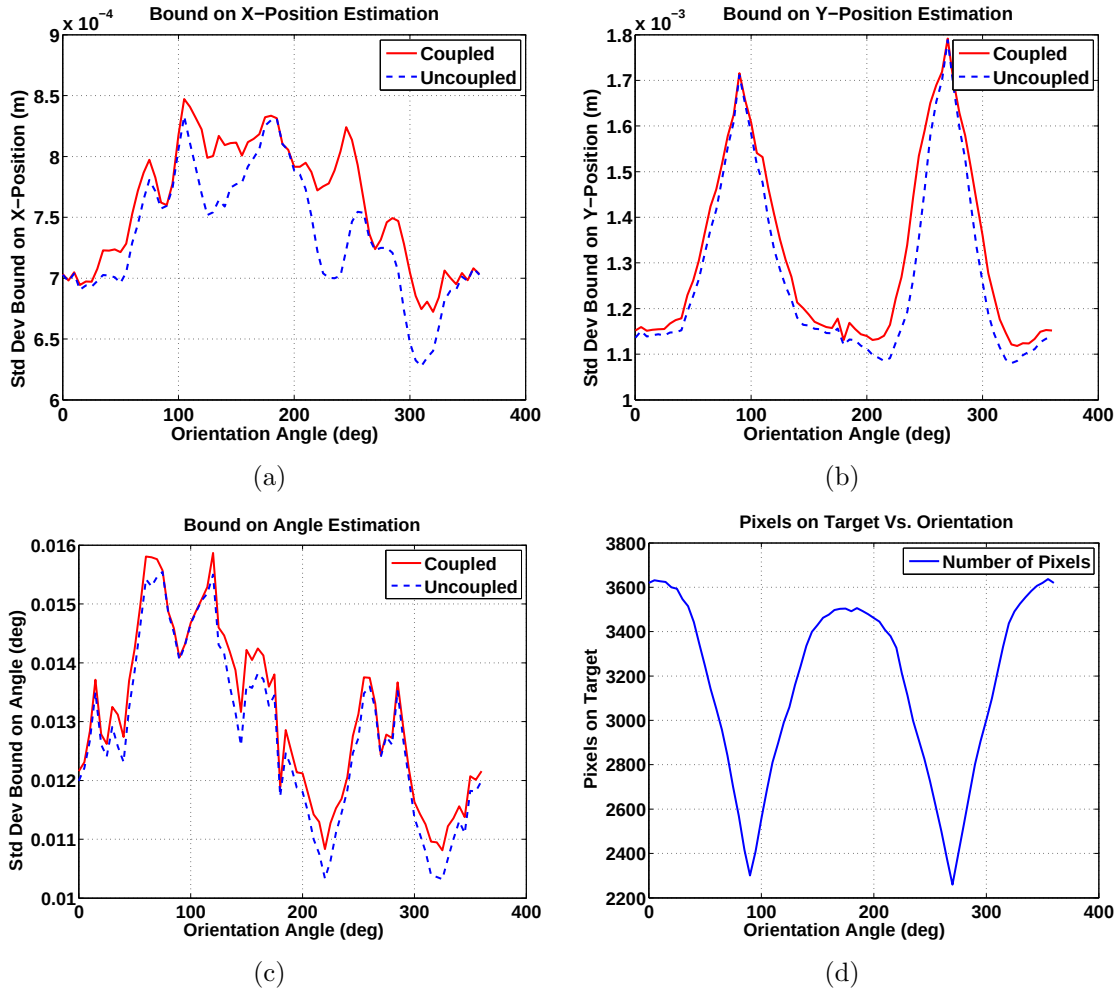
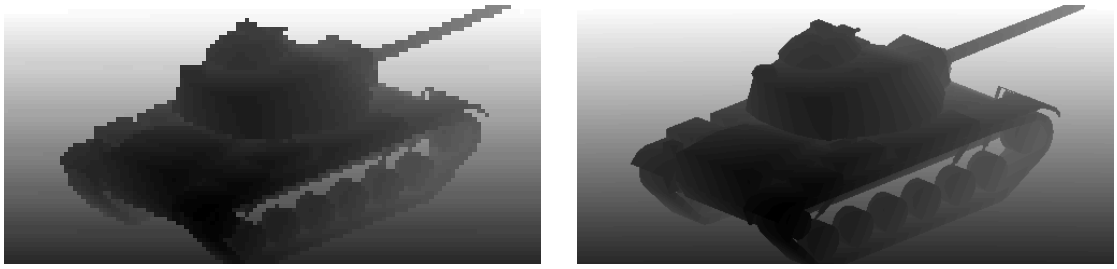


Figure 36: Cramér-Rao bounds on estimating (a) x-position, (b) y-position, and (c) orientation angle for a single M60 tank with respect to the current orientation angle of the tank. Figure (d) shows how the number of pixels on target changes with orientation angle.

5.5.3 CRLB and Image Resolution

The LADAR model employed previously acquires range imagery with a resolution of 125×60 pixels. The section discusses how the bounds are affected by changes to image resolution. We repeated the experiment defined in the previous section, where we determined the bound on y-position estimation with respect to orientation angle, except this time, we used an image resolution of 500×240 pixels. Sample images from this study are shown in Figure 37. Plots from the experiment can be seen in Figure 38. The general trends in both plots remain unchanged, as can be observed by the presence of strong peaks at the orientation angles where the target is facing directly towards or away from the LADAR sensor. The values of the standard deviations, however, differ by more than a factor of four. This is to be expected, since the higher resolution case has more pixels while the noise-per-pixel remained the same. An interesting future experiment might be to change the standard deviation between the two cases to keep the overall signal-to-noise relatively constant. Another major difference between the charts can be seen in the ripples in the CRLB curve for the lower resolution images.

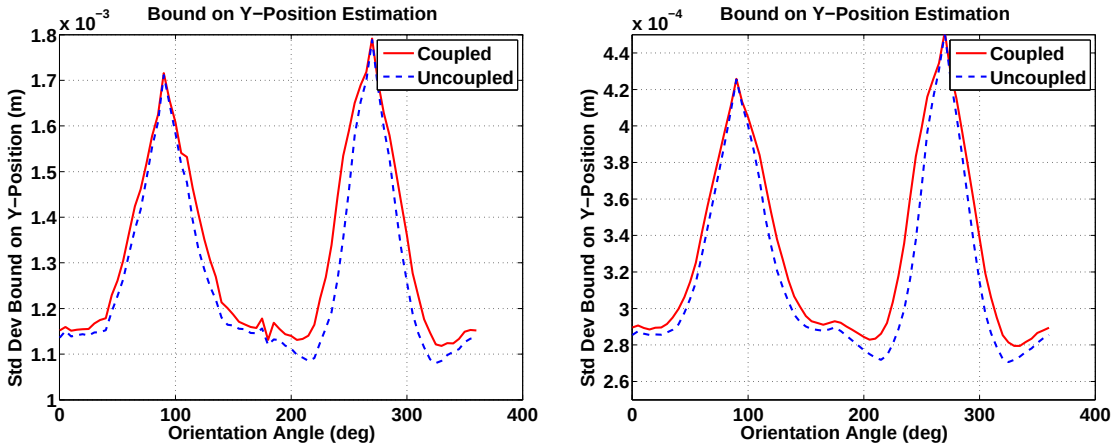


(a) (b)

Figure 37: Images of M60 tanks at different resolutions. Image (a) is 125×60 pixels and image (b) is 500×240 pixels.

5.5.4 CRLB and Derivative Stepsizes

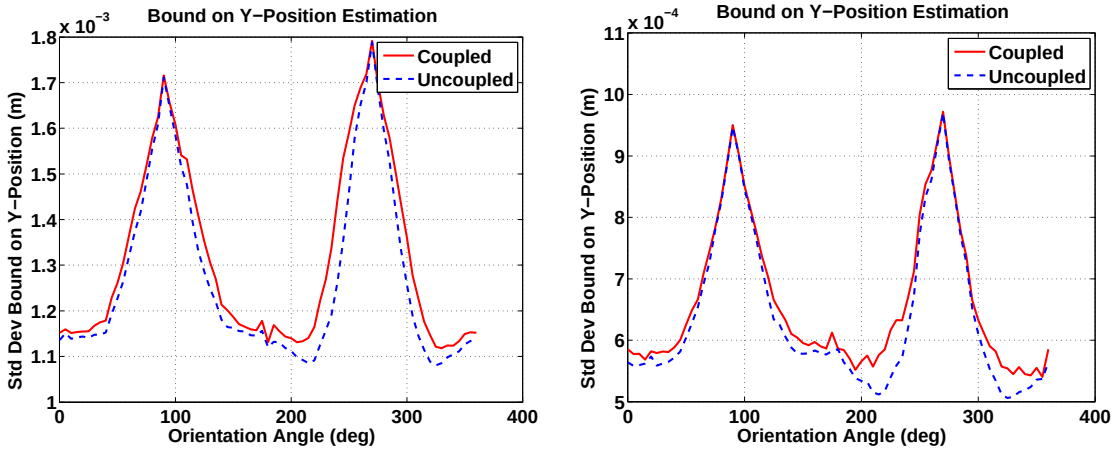
One significant problem encountered in this study is the choice of stepsize for the derivatives used to compute the Fisher information matrix. The computation of these



(a) (b)

Figure 38: Cramér-Rao bounds on estimating y-position for a single M60 tank with respect to the current orientation angle of the tank. Graph (a) was computed using a resolution of 125×60 pixels and graph (b) was computed using a resolution of 500×240 pixels.

derivatives is tricky in that there is no clear analytic means to do so. The plots shown in Figure 39 provide an illustrative example. The stepsizes chosen for the CRLB computations in Figure 39(b) are one-quarter the size of those chosen for the computations in Figure 39(a). One noteworthy difference is seen in the absolute magnitudes of the bounds themselves. The bounds computed using the smaller derivative stepsizes are, on average, slightly lower than those computed with the larger derivative stepsizes. A second difference between the two plots is seen in the subtle difference between the coupled and uncoupled curves; specifically, the bounds computed with the larger stepsizes appear to have larger "gaps" between the curves. Intuitively, it makes sense that the smaller stepsizes should offer the more precise derivative computation, but given the nature of the rendering process and the discretization of resulting images, this may not be so. Further study is needed to determine if ideal stepsizes can be chosen, or if a better approximation than the first difference may be found. Again, the pixel-based diffusion results from Chapter 3 may be applied to the problem of automatically determining position and scale-specific stepsizes to use in the finite different approximation in future work.



(a) (b)

Figure 39: Cramér-Rao bounds on estimating y-position for a single M60 tank with respect to the current orientation angle of the tank. Graph (a) was computed using a derivative stepsize four times the size of that used for the computations in graph (b).

5.6 Monte Carlo Trial Experiments

To see how the theoretical Cramér-Rao bounds might compare with results from an actual ATR algorithm, we have chosen a few special cases from the CRLB experiments and compared the bounds to those simulated from a set of Monte Carlo trials using repeated simulation under specific conditions and multiple noise realizations. Since our CRLB results represent bounds on the standard deviation error of pose estimation, our Monte Carlo results will average the error over multiple simulations to form a curve representing a sample standard deviation.

We compare our theoretical bound on pose parameter estimation estimation to empirical performance by conducting two types of Monte Carlo experiments: one where we vary the sensor noise, plotting the noise level along the horizontal axis, and another where we vary target orientation angle, plotting that angle along the horizontal axis. The position coordinates, target type, and viewing parameters are held constant. Varying the sensor noise allows us to find a range of system parameters for the orientation experiment because, in practice, certain SNR ranges provide better

conditions for Monte Carlo experiments. If the sensor noise is too low, the error in pose parameter estimation is nearly negligible, but extremely high sensor noise prevents the pose estimation algorithm from achieving any accuracy whatsoever. In the case of a Gaussian distribution, the CRLB changes linearly with respect to changes in noise variance, regardless of how minor the change. In our simulations, we find that noise variances that are too small will not affect the parameter estimation algorithm significantly, while noise variances that are too large will cause the estimation algorithm to simply fail. Once we have found appropriate sensor parameters to generate an appropriate amount of noise, we may see how the estimated bounds compare to the theoretical ones while varying the orientation angle.

To make the calculations easier, we use a simplified LADAR sensor model where noise is modeled as a Gaussian distribution with a mean given by the true range and variance given by an arbitrary value. This model differs from the LADAR sensor likelihood equation (7) discussed in Section 1.3.2 because the noise variance term $\sigma^2(n)$ is no longer a function of the pixel index, but a constant value.

For these simulations, an M60 tank is placed on a ground plane 1 km from a simulated range sensor. An image is formed by scanning over angles of 25 mrad horizontally and 12 mrad vertically. We sample 60 times over the vertical sweep angle and 125 times over the horizontal sweep angle so that the dimensions of the resulting image are 60×125 pixels. For these experiments, we estimate the pose parameters using the pattern-theoretic ATR pixel-based diffusion process discussed in Chapter 3. We use this simplified pose estimator, instead of the full jump-diffusion process discussed in Chapter 2, to reduce the number of computations required for each pose estimate and therefore significantly reduce the simulation run time. Before each estimate, we create a synthetic LADAR scene with the M60 in a known position and we corrupt the scene with Gaussian noise of a known variance. We initialize the pose estimator to a randomly chosen value so that there is some overlap between

elements of the target at that position and elements of the target at the true position. The reason for this initial overlap is that the diffusion process requires a starting "guess" that is close to the true target position because it is meant to act as a way to refine the pose parameters after a target has been hypothesized by the ATR algorithm. The number of iterations was heuristically chosen in a way that allows the estimator to converge as close as possible, given the current noise conditions, to the true pose parameter estimates.

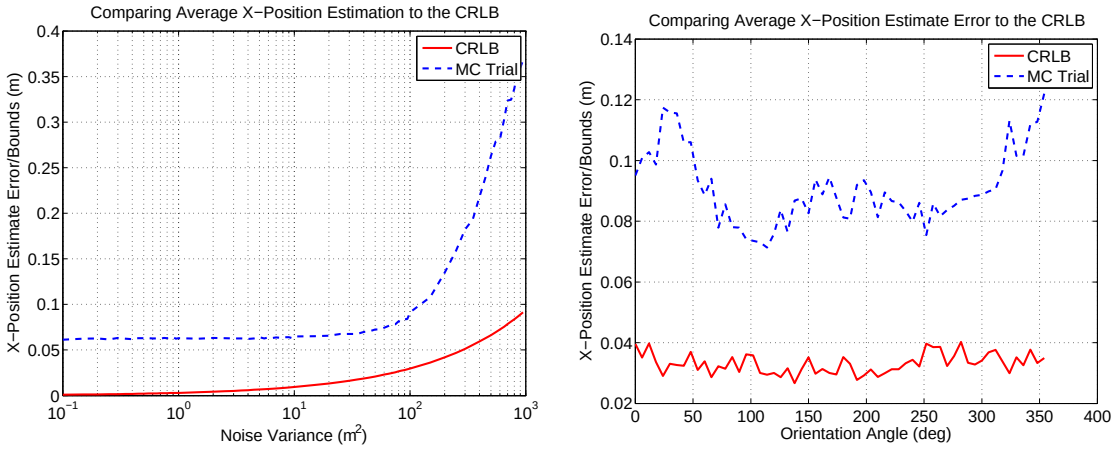
5.6.1 CRLB and Average Standard Deviation Comparisons

We calculated each empirical standard deviation estimate from 1000 pose parameter estimates using the aforementioned pixel-based diffusion process. For each pose parameter, we computed two curves: one representing how the pose estimation accuracy changes as the image noise increases and another representing how the pose estimation accuracy changes with orientation angle at a given noise level. For the orientation experiment, we chose a noise level that is high enough to introduce pose estimation error but low enough so that the diffusion algorithm can still reasonably estimate the parameter in question. This noise level is chosen at the "knee" point of the Monte Carlo curve.

Figure 40 compares the the CRLB for estimating the x-position with the Monte Carlo estimation results over a range of noise levels and orientation angles. As expected, the Monte Carlo curve initially has a slight upward slope as the noise level increases. During this interval, the noise induces nearly constant error in the pose estimator. Since the diffusion is discretized on pixel boundaries, there is a fundamental limit to how close it can converge to the "true" pose parameter value. If the diffusion were modified to have sub-pixel accuracy, the Monte Carlo curve might be closer to the CRLB. The fact that the slope of the Monte Carlo curve is near zero while the noise level is in this range indicates that this amount of noise does not affect the

diffusion algorithm. It still reaches the same point as if there were no noise. At the point of the "knee" in the curve, the noise begins to affect the estimation accuracy. The estimation error increases logarithmically from this point.

The noise variance at the "knee" point is used for the x-position estimation performance vs. true orientation angle experiment. Earlier, we found the CRLB to indicate that there was no clear relationship between the orientation of the tank and the performance of x-position estimation. The Monte Carlo curve appears to behave similarly, not presenting a well-defined relationship to the orientation angle, but consistently above the CRLB as we would expect.

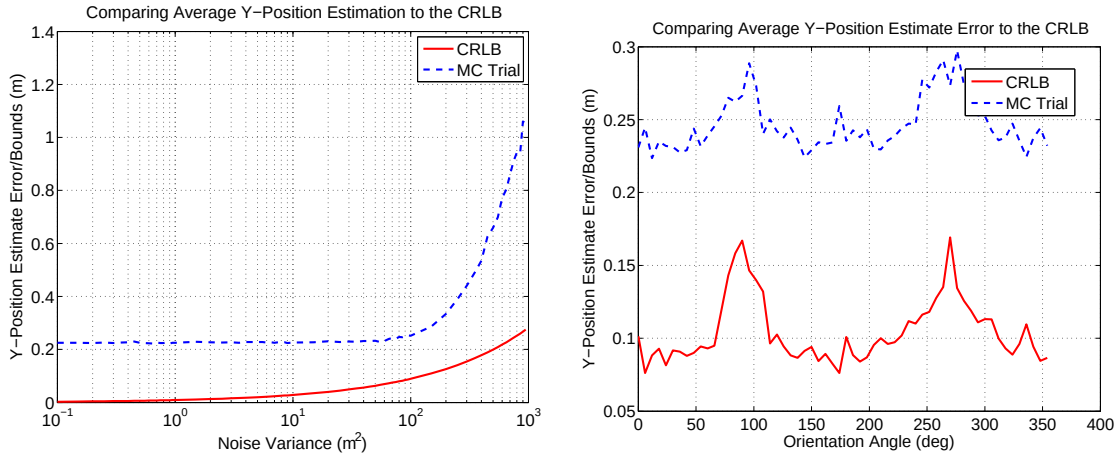


(a) (b)

Figure 40: Cramér-Rao bounds on estimating x position compared to the average estimation error from 1000 Monte Carlo trials. The average estimation error is computed as a sample standard deviation over 1000 estimates at 72 noise variance values in plot (a) and 60 orientation angles in plot (b).

Figure 41 compares the CRLB for estimating the y-position with Monte Carlo results. The shape of the curve, and its relationship to the CRLB, is nearly identical to the x-position estimation case. For the experiment varying true orientation, shown in Figure 41(b), the height of the Monte Carlo curve also has peaks at the locations where the target is pointing directly towards or directly away from the range sensor, confirming the conclusion from Section 5.5.2, which suggested the CRLB looked this way because the scene geometry presents the least amount of useful information when

estimating y-position when the targets face these directions.

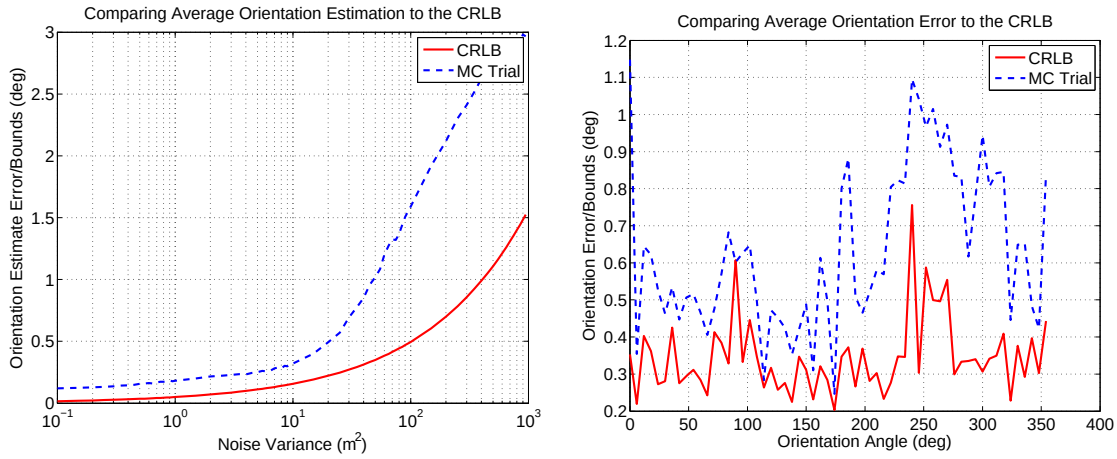


(a) (b)

Figure 41: Cramér-Rao bounds on estimating y-position compared to the average estimation error from 1000 Monte Carlo trials. The average estimation error is computed as a sample standard deviation over 1000 estimates at 72 noise variance values in plot (a) and 60 orientation angles in plot (b).

Figure 42 compares the CRLB for estimating orientation angle θ with Monte Carlo results. The noise variance vs. estimation bounds/error curves differ from the analogous x and y-position curves in that the Monte Carlo curve is much closer to the CRLB and the "knee" point occurs at a noise level that is an order of magnitude lower. Interestingly, the experiment plotting true orientation along the horizontal axis, shown in Figure 42(b), shows a Monte Carlo envelope much closer to the CRLB, just barely above the CRLB in some instances. Once again, there appears to be no immediately obvious relationship between the orientation angle and the corresponding pose estimation error.

Most of this dissertation is written from a Bayesian viewpoint, in which parameters have prior distributions and diffusion-like algorithms are used to draw random samples from the Bayesian posterior distribution. Although there is a Bayesian version of the CRLB [61], which includes an expectation over the parameters as well as the data, the version of the CRLB used here, which is a function of the "true" parameters, is



(a) (b)

Figure 42: Cramér-Rao bounds on estimating the orientation angle θ compared to the average estimation error from 1000 Monte Carlo trials. The average estimation error is computed as a sample standard deviation over 1000 estimates at 72 noise variance values in plot (a) and 60 orientation angles in plot (b).

more congruent with a frequentist viewpoint. Maximum-likelihood parameter estimates can be thought of as equivalent to Bayesian MAP estimates with uniform prior probabilities. Technically, the diffusion-like algorithm used in the Monte Carlo trial experiments produces sequences of random samples that could be averaged to form a minimum mean-squared error (MMSE) estimate. Alternatively, the sample with the highest posterior distribution could be tracked and considered the MAP estimate. In practice, the posterior tends to be so heavily peaked at the MAP estimate that diffusion algorithms, in execution, resemble a deterministic search for a maximum. In a finite number of iterations, there is no guarantee that the "true" maximum has in fact been reached, and the algorithm could effectively become "stuck" in a local minimum, even with a random perturbation term. The 1000 Monte Carlo runs might include a few such cases.

5.7 Conclusion

This chapter discussed accuracy bounds on target pose estimation. We derived the CRLB from a simplified version of the sensor likelihood function we use for LADAR

sensor data. To calculate the CRLB, we employed an approximation that reduced the computational complexity of determining derivative computations of target parameters in two-dimensional range imagery. We showed that the shape of the CRLB correlates well with our intuitive understanding of how pose estimation accuracy may be affected by viewing geometry and target pose. We briefly explored the effect of sensor resolution on CRLB computation and how the choice of derivative stepsizes can affect the derivative computation in the CRLB. Finally, we showed that Monte Carlo trials produced results that corresponded with our CLRB calculations.

While most ATR algorithms are designed to be invariant with respect to position and orientation, our information-theoretic perspective illustrates that even algorithms that do not explicitly involve the estimation of such nuisance parameters may still be affected by them [23].

In general, derivation of the CRLB only requires the sensor likelihood function be specified and differentiable with respect to the pose parameters of interest. Since the form of the LADAR sensor likelihood is similar to the form of the FLIR sensor likelihood, the techniques of this chapter could also be applied to parameter estimation with FLIR data, not just LADAR data.

CHAPTER VI

CONCLUSION

6.1 Contributions

The contributions presented in this dissertation are summarized as follows:

- Development of infrared image calibration techniques that function in the pattern-theoretic framework;
- Incorporation of thermal state inference and in the jump-diffusion process for pattern-theoretic ATR;
- Transition from a gradient ascent-based diffusion process to a more intuitive pixel-based diffusion process;
- Application of Cramér-Rao bounds to the determination of pose estimation accuracy in laser radar imagery and verification of those bounds through Monte Carlo simulation;
- Development tools for synthetic scene creation to be used by others for research in automatic target recognition and target recognition performance.

6.2 Future Work

Some of the contributions of the dissertation can be readily applied to the jump-diffusion ATR algorithm. Future work can include implementing the pixel-based diffusion algorithm into the full ATR algorithm for FLIR and LADAR data. For FLIR data, this can be performed in stages to test its efficacy assuming different levels of certainly about the thermal characteristics of the scene. For instance, the jump-diffusion algorithm with pixel-based diffusion can be used on a scene with known

thermal content, a scene with unknown, but calibrated, thermal content, and a scene with unknown and uncalibrated thermal content.

Lastly, we mention some pattern-theoretic research topics for future investigation.

6.2.1 Rapid Target Detection for Arbitrary Sensors

One of the more significant issues with the jump-diffusion algorithms described in this dissertation for image-based ATR occurs in the initial target detection stage. Although targets can be found over time via the birth strategy [40, 39] if the viewing grid is sampled finely enough, this process does not make intelligent use of the available information. Guessing locations to search and then rendering fully detailed CAD models at those locations is computationally expensive and, in practice, does not guarantee that all targets will be detected. A good target detection algorithm should consider the quality of a detection, that is, how well hypothesized target region covers the actual target within the image. For pattern-theoretic ATR, the diffusion process should account for these situations, but it sometimes has difficultly refining the estimated pose parameters. A simpler strategy for rapid target detection could use rectangular regions to represent potential targets. Creating hypothesized targets in this way, using the same sensor likelihood function to evaluate our hypothesis, could greatly reduce the computational complexity of the algorithm because we no longer have to render fully detailed CAD models at each potential target location. New types of jump moves could switch between the coarse rectangular representations and specific target models.

6.2.2 Metrics for the Rejection of False Targets

The pattern-theoretic ATR algorithms discussed in this dissertation distinguish between only background and known target types. This means that over time, clutter and unknown target types within the sensor's field of view may be erroneously classified as known targets with a certain pose. Future efforts can examine different types

of penalties [42, 26, 51, 63] that can be subtracted from the likelihood function to reduce these false alarms.

6.2.2.1 False Targets

With the incorporation of the estimation of thermal states of targets, our jump-diffusion algorithm suffers from a potential increase in false alarms when initially detecting targets. This can be attributed to the amount of variability that our current eigentank model is capable of capturing, which includes thermal states that are generally not attainable by actual targets. If we are rendering a target model over a segment of the data image that contains a target of the same type, and if the pose parameters match up correctly, then the inferred thermal states should also closely match the true values since the intensity regions will overlap properly. If any of these conditions are not met, then the computation of the expansion coefficients will either include matching the hypothesized target's intensity regions with the wrong regions of corresponding target in the data or matching intensity regions with background. If the background includes clutter that contains thermal characteristics similar to those of known targets, then the algorithm may choose a thermal profile that blends into the background. If this happens during a birth move in the jump-diffusion process, a *phantom target* can appear in the hypothesized scene. Since the likelihoods tend to increase with the existence of these phantom targets, it becomes more difficult for the jump-diffusion algorithm to remove them later on through an iteration of the death process. These phantom and misaligned targets can have thermal profiles that make little physical sense (e.g., the exhaust may be cooler than the rest of the tank) because of the symmetry of our PCA eigentank representations around the coefficient means.

An example of this can be seen in Figure 43. Here, we simply computed expansion coefficients at four different positions over an infrared image that contained no

targets. The purpose was to see which facets of the resulting targets had thermal characteristics similar to the background data. As seen in the image, some target features appear as they would on a typical data set while other features blend into the background to the point where the tank almost disappears.

Figure 43: Tanks with radiance profiles derived from background emissions.

6.2.3 New Representations of Thermal State

Although the PCA analysis used to create the eigentanks appears to be a useful technique for representing thermal state, no work has been done to compare this with other dimensionality reduction techniques. From observing the estimated thermal states during the jump-diffusion process, we see that the eigentank model suffers by allowing target signatures to be predicted in a way that would be physically impossible. The eigentank research was initially motivated by the positive results obtained from working with eigenfaces for face recognition. Methods like independent component analysis have also been applied to facial recognition [14], which could bear fruit in representing thermal states as well. In addition, there have been improvements to software-based infrared signature prediction and modeling. Software packages like Irma [55], infrared prediction software developed by Northrop Grumman, and Muses

[34, 53], the successor to the PRISM software, also developed by ThermoAnalytics, can be used to create expanded representations of thermal state beyond what our Prism models provided.

More importantly, future work should consider using infrared prediction rather than predefined profiles to reduce the reliance on training data. Capturing the underlying physics behind infrared signature prediction and applying it to the sensor likelihood formulation would be a better fit to the pattern-theoretic framework. This is a nontrivial task, requiring the complete modeling of all parameters and physical processes that are needed to produce all possible thermal profiles for a given target. Currently, the scale of a probabilistic model with that much complexity would present a number of implementation challenges unless significant computing resources are available. As computer power increases, these processing constraints can be relaxed over time. Another tactic beyond raw computing power would be to determine which physical process parameters yield the greatest thermal variability.

APPENDIX A

SYNTHETIC SCENE GENERATION TOOLS

A.1 Introduction

We performed a significant part of this dissertation research using a set of synthetic scene generation tools for LADAR and FLIR data that we created as part of another effort. These tools, and the knowledge gained from their creation, contributed greatly to the Pattern-Theoretic ATR advances presented in this dissertation. This appendix describes these tools and how they can be used to create synthetic LADAR range imagery and three-dimensional point cloud datasets within MATLAB. Scenes are synthesized using one or more faceted models at user-specified coordinates, orientation angles, and sizes. The main components of the tool are a graphical user interface (GUI) and a simple application programming interface (API). The GUI aids scene creation by facilitating object placement, specifying the ground plane, defining the viewing perspective, and adjusting model parameters. The API allows users to write custom MATLAB scripts analyze their own data sets. For LADAR, range, and point cloud data, the tools currently support faceted models in PRISM, 3D Studio, STereoLithography (STL), and Princeton Shape Benchmark [57] object file formats. For FLIR data, the tools only support the PRISM faceted model file format and associated infrared prediction modeling metadata.

A.2 Scene Creation Overview

This section provides a high-level overview of the simulator and using it to define and render scenery.

A.2.1 Parts of a Scene

The simulator software allow users to create synthetic data sets (range images, point clouds, and infrared scenes) from predefined, three-dimensional scenes. The tool starts by setting up a three-dimensional world view of a particular scene in a space with axes x, y, and z. Axes x and y represent coordinates along the ground, or any plane at some given height (as in one of the standard three-dimensional coordinate systems used in computer graphics). The z-coordinate represents the height at particular x and y values. If we consider the ground, then all z values will be zero. A viewing sensor is usually positioned at some point above the $z = 0$ plane, pointing at some predefined point in the world. The position of the sensor may be defined in rectangular x, y, and z coordinates, or in terms of range, azimuth, and elevation spherical coordinates based on the "look at" point. A flat, rectangular ground plane of any desired size can be placed at the coordinates where $z = 0$ to give the appearance that objects in the scene are resting on a surface.

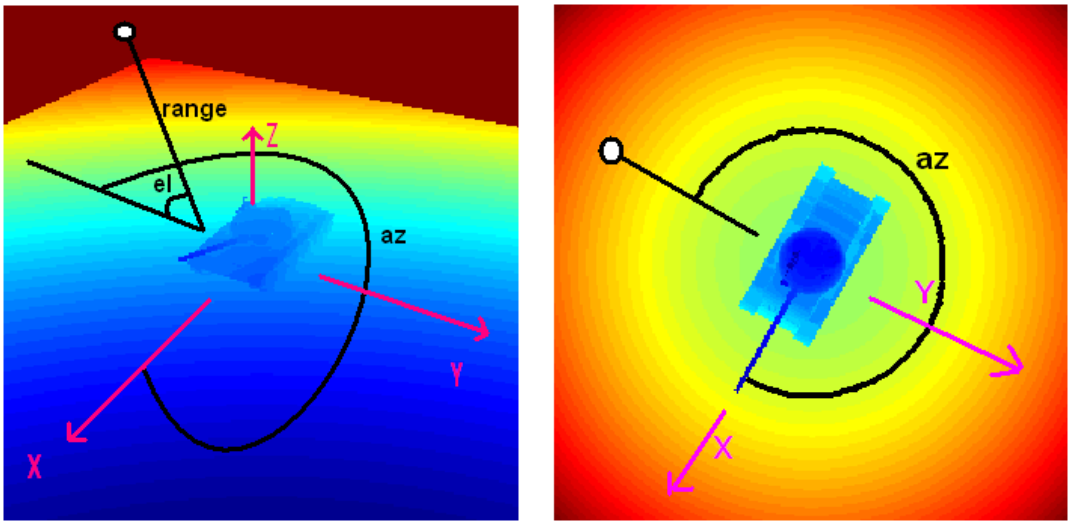
The units of the world coordinates are not explicitly defined and may represent anything. For example, suppose the user wants to view a tank from 50 meters away, but the faceted model of the tank is defined in millimeters with dimensions $6367 \times 3122 \times 2943$. Specifying 50 as the range will result in an appearance that may be interpreted in two different ways: (1) the tank is 6367mm $\times$3122mm $\times$2943mm and the sensor is 55mm away, or (2) the tank is 6367m $\times$3122m $\times$2943m and the sensor is 50m away. Neither interpretation is useful; without scaling the values, the distance from the target and the dimensions of the tank are considered to be the same units. To have all objects in the scene drawn appropriately, the units of the tank must be converted to meters by scaling by 1/1000, or the units for the sensor's range from the target must be specified in millimeters.

A.2.2 Setting the View

The viewing area and resolution of the resulting imagery are set via the field of view and data dimension parameters. Field of view (FOV) is defined in the horizontal and vertical directions, relative to the position of the sensor, in units of degrees. In essence, the sensor scans +FOV/2 and -FOV/2 degrees in both directions. The number of pixels in the image is determined by the data dimension parameters (one can think of this parameter as the angular sampling that occurs as the sensor scans over field of view). Square images are created by setting the two field of view angles to the same value, while rectangular images result from one field of view angle being larger than the other, regardless of the pixel dimensions. The sensor's position can be specified in world or spherical coordinates. To use world coordinates, simply set the $[x, y, z]$ parameter vector. For spherical, set the $[\rho, \theta, \phi]$ coordinates, representing range, azimuth, and elevation. An illustration of the view and coordinate system can be seen in Figure 44. Range is taken to be the distance between the sensor and the "look at" coordinate, also specified as a $[x, y, z]$ parameter vector. The minimum and maximum viewable ranges must also be set using the appropriate parameters. When viewing point clouds, the data dimension and field of view angles define the scene from which the points will be extracted.

A.2.3 Adding Targets

Scenes may contain any number of objects (hereafter often referred to as "targets") in a variety of positions and orientations. Target geometries are obtained by reading in CAD model descriptions. Targets are placed at specified (x, y, z) coordinates and rotated by some angle around the three axes. Targets may be rescaled if the units used to define the target's CAD model are undesirable.



(a) (b)

Figure 44: A standard (a) and overhead (b) view of a scene's layout. The sensor location is represented by the white circle.

A.3 Graphical User Interfaces

This section reviews the different graphical user interfaces (GUIs) developed for the LADAR Simulator.

A.3.1 Main LADAR Simulator GUI

To run the LADAR Simulator GUI, type `ladar_sim` in the MATLAB prompt. The GUI will prompt the user for a configuration file to load. This file contains information for setting up a scene. An example configuration file can be found in `ladar_simulator/config/sample.config`. If you wish to load default configuration settings, just press the `Cancel` button in the dialog box.

Next, another dialog box will appear. This box asks the user to load a model library file. This file contains structural information about the models referenced in the configuration file. If a configuration file was selected, a predetermined model filename will be chosen and the user must locate that file. If no configuration file was selected, the user is free to specify any available model library file. The default model library file is `ladar_simulator/sample.ml`. If no model libraries are available,

the user may create one by pressing the `Cancel` button and using the Model Library Creation GUI. For instructions on how to use this GUI, see Section A.3.3. For the LADAR Simulator GUI to run, a model library must be selected or created.

Once the model library has been chosen, the LADAR Simulator GUI will appear (see Figure 45). The GUI contains all components necessary to create a scene. The axes on the right side of the window display the rendered configuration or point cloud. The type of scene is selected from three possible choices using the pop-up menu below the scene axes: `Full Range`, `Point Cloud`, or `Depth Buffer`. For a detailed description of these scene types, see Section A.5.3. By default, the color mapping used for two-dimensional data sets is determined automatically by the minimum and maximum values in the scene. This can be overridden by marking the `Colormap` check box and selecting the minimum and maximum values for color map scaling. The user can add or remove a ground plane to the scene by selecting the `Ground Plane` check box. The dimensions of the ground plane can be defined by pressing the adjacent `Options` button. For details about setting up a ground plane, see Section A.3.5. Gaussian distributed noise of a chosen variance may be added to the scene by selecting the `Noise` check box. The noise is centered on the range values (or coordinates in the case of point cloud data).

The *Target List* listbox shows targets that have been added to the current configuration. The text fields below the list box are used to set target parameters. Targets may be added, removed, or updated by pressing the corresponding buttons next to the `Target List` list box. Targets may also be added with the mouse by left-clicking on the desired location in the scene. When you select an item in the listbox, the corresponding target parameters will appear in the text fields. Users may also update existing targets by pressing the `Enter` key after changing one of the text field values. Scene configurations may be loaded from configuration files by pressing the `Load` button. The `Save` button will save a configuration to a file. The adjustable

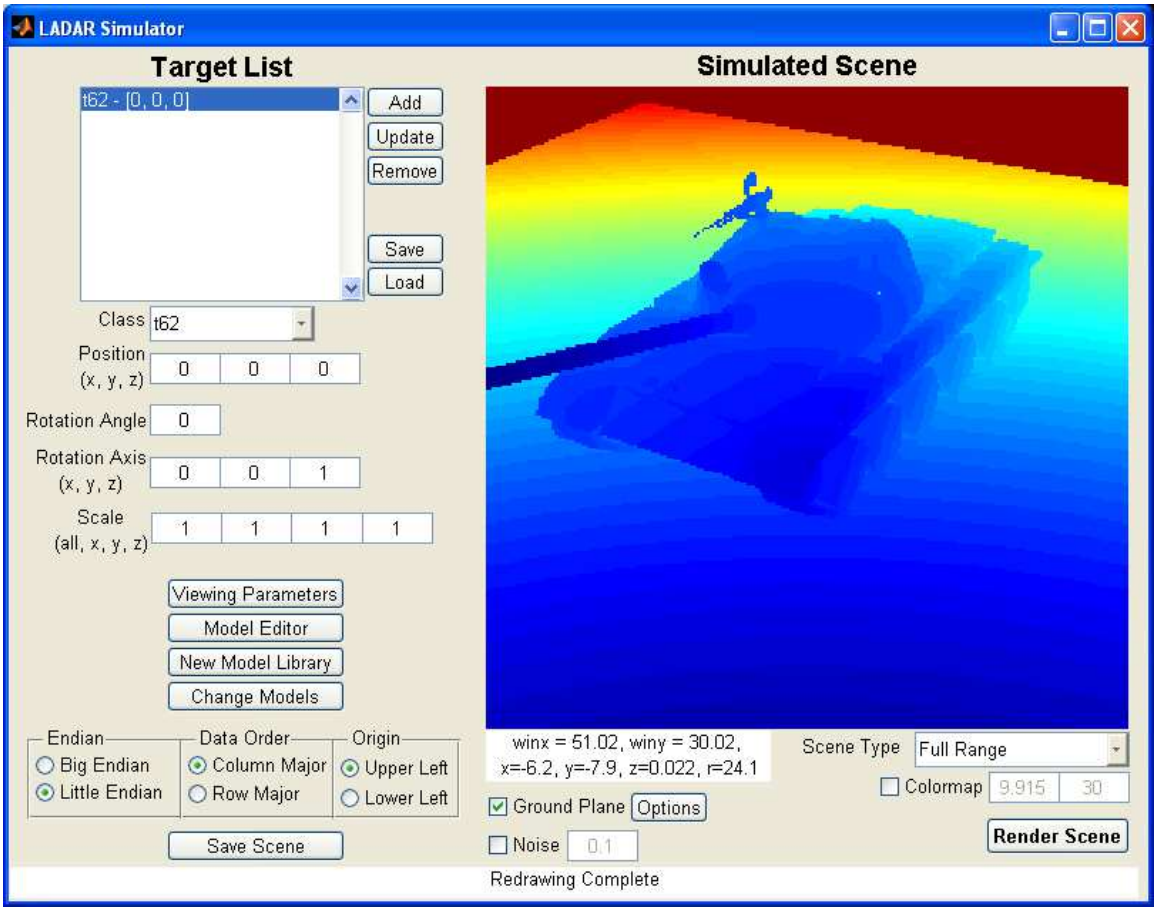


Figure 45: Main LADAR Simulator GUI window.

target parameters are as follows:

- Position: the location of the target in the scene. x and y are ground coordinates, while z is the coordinate above or below the ground plane. In most situations, z will be set to zero.
- Rotation Angle: an angle rotation in degrees starting from the positive x-axis and moving counter clockwise.
- Rotation Axis: axis about which to perform rotations. To perform rotations about z-axis, this should be set to $[0, 0, 1]$.
- Scale: stretches or shrinks a target by multiplying the scale by the coordinates used to define the target.

From within the main LADAR Simulator GUI, other GUIs may be launched. To change the viewing perspective and associated parameters, press the `View Settings` button (see Section A.3.2). To adjust CAD model parameters, press the `Model Editor` button (see Section A.3.4). To create new model libraries, press the `New Model Library` button (see Section A.3.3). To change the current model library, press the `Change Models` button.

To save range images or point clouds, users can use the corresponding button and radio toggles. When saving data sets, it is best to remember the format used because that information will be necessary for correctly reloading datasets. By default, data is saved in the native machine format, column major, with the upper left corner of the image set as the origin.

A.3.2 View Settings GUI

The View Settings GUI facilitates the adjustment of scene viewing parameters (see Figure 46). The text fields of the GUI are as follows:

- Field of View: two fields representing the vertical and horizontal scanning angles used to determine the viewing area. The angles are interpreted as + or - fov/2 from the current "look at" position in either direction.

- Position (cartesian): $[x, y, z]$ coordinates representing the sensor location. When the cartesian coordinates are changed, the spherical coordinates are adjusted automatically.

- Position (spherical): $[\rho, \theta, \phi]$ coordinates representing the range, azimuth, and elevation from the "look at" point in the scene. When the spherical coordinates are changed, the cartesian coordinates are adjusted automatically.

- Look At: cartesian coordinates representing the point to which the sensor is pointing. When the "look at" point is changed, the spherical coordinates are adjusted automatically. It is assumed that changing the "look at" point does not reflect a change to the sensor position.

- Min/Max Range: two fields representing the placement of the clipping planes relative to the sensor location. Targets or part of the ground plane placed beyond these range limits will not appear in the rendered scene. Pixel values in range images that represent scene elements beyond the range limits will be clipped to the minimum and maximum range values.

- Dimensions: two fields representing the vertical and horizontal (number of rows versus number of columns) pixels in range images. For point clouds, these fields represent the number of samples used to extract point cloud information (in the event that all samples are valid points, then the product of these two fields is the number of points in the point cloud). When combined with the field of view angles, these fields can be thought of as the angular scanning sampling in the vertical and horizontal directions.

- Up Vector: supplies the rendering system with a vector orientation for "up". In most cases, these may be set to $[0, 0, 1]$, representing the positive z-axis as the up direction. If the camera is pointed straight down, then a new up direction must be chosen so that the vector has a projection along the xy-plane.

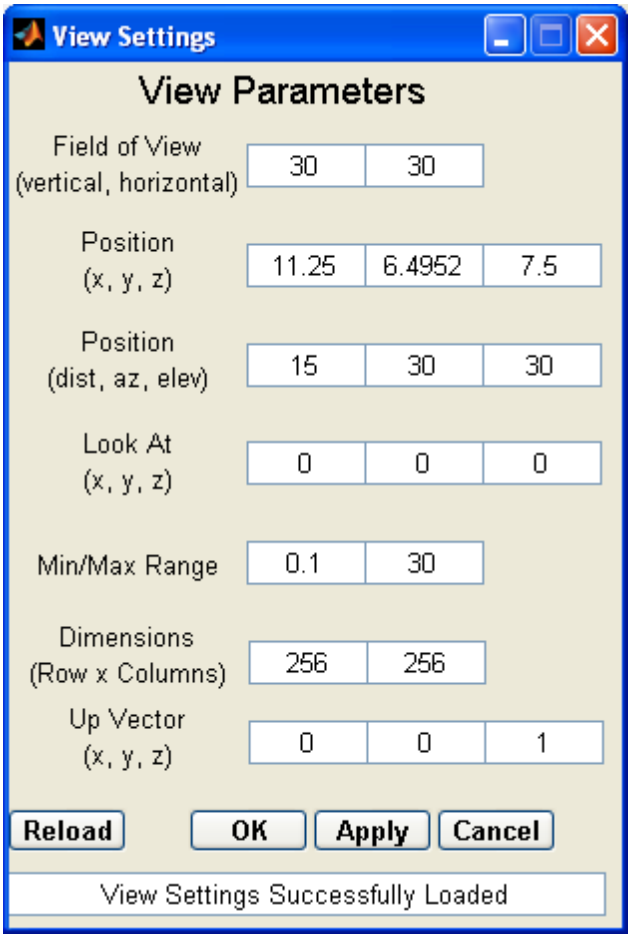


Figure 46: View settings GUI window.

When changing fields, the GUI will perform a general validity check for the most recently used parameter and revert to the original value if the current value is found to be invalid. Pressing the `Reload` button will reset the GUI to its original state after it was first launched. Pressing the `OK` button will save the view settings, close the View Settings GUI, and redraw the scene using the new settings. Pressing the `Apply` button will save the current view settings and redraw the scene. Pressing the `Cancel`

button will close the View Settings GUI and discard all changes since the last time the `Apply` button was pressed. Pressing the `Enter/Return` key after editing a text field simulates pressing the `Apply` button.

A.3.3 New Model Library GUI

The LADAR Simulator GUI allows users to create their own model library files consisting of references to CAD models in supported model file formats and a set of adjustable parameters. When the "New Model Library" button in the LADAR Simulator GUI window is pressed, a dialog box will appear that allows the user to select the name and path of the model library file to be created (see Figure 47). The file must be saved to a directory in, or above, where the CAD model files are stored. Once the file has been specified, the New Model Library GUI will appear, as shown in Figure 48.

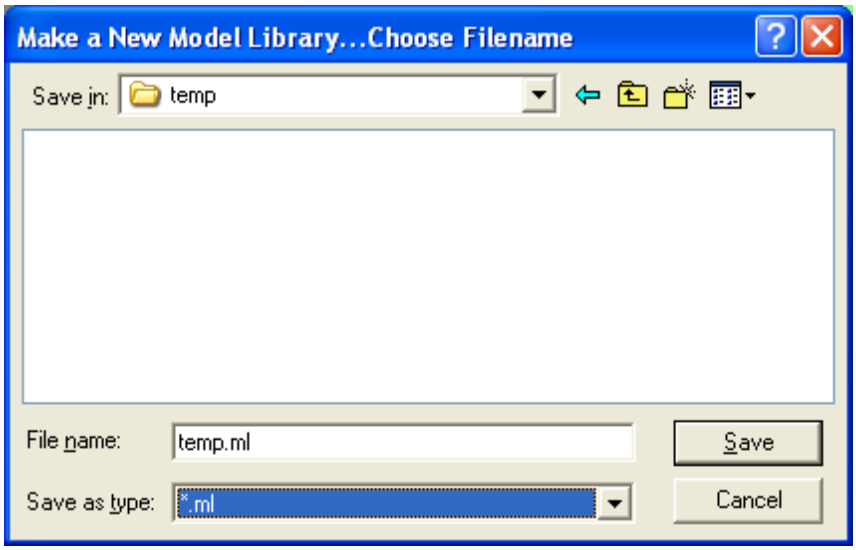


Figure 47: Choose the filename and path for the model library file.

The New Model Library GUI is an interface used to create a model library file with chosen model CAD files (see Figure 48).

The listboxes and fields are defined as follows:

- Model List: listbox that displays the model files currently in the library. When

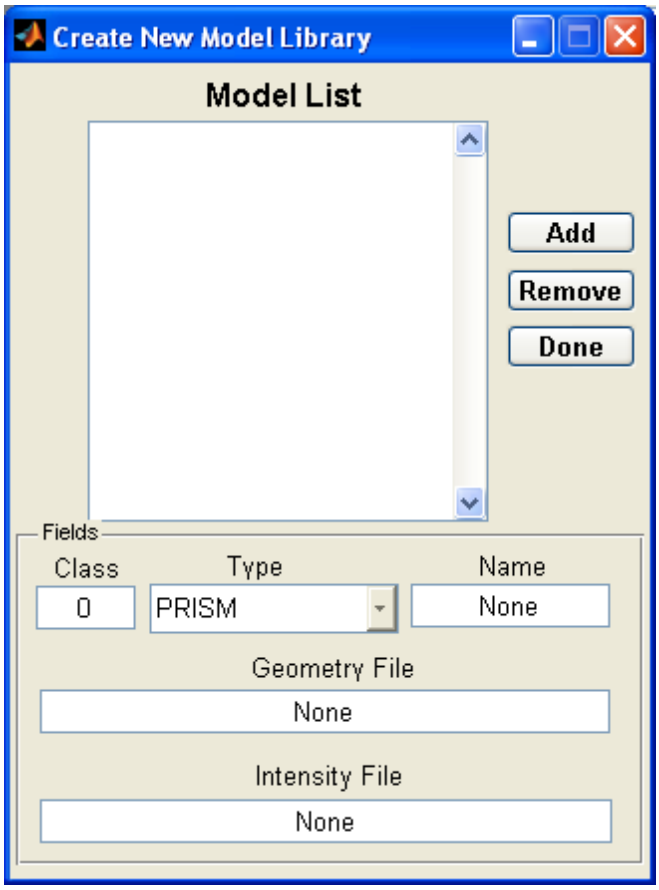


Figure 48: Empty model library creation GUI.

a model is selected, the corresponding parameters will be shown in the GUI text fields.

- Class: a unique non-negative integer used to identify a model in the library.
- Type: a pop-up menu that allows you to choose among CAD file types. These include STL, PRISM, 3DS, and OFF.
- Name: a text string identifier for the model
- Geometry File: path and location of the geometry CAD model file relative to the location of the model library file.
- Intensity File: the same file and path as in the Geometry File field, unless working with PRISM models. In that case, use the radiance filename.

For more information about the meaning of these fields, see Section A.5.2.3.

To add model files to the library, press the `Add` button. A file dialog like the one shown in Figure 49 will appear. Navigate to the subfolder with the geometry files and select the one(s) you wish to add to the model library. After clicking `Open`, the Model List listbox will populate with the chosen models and the corresponding text fields can be seen by clicking on a model in the list (see Figure 50). By default, the classes are numbered sequentially in the same order in which the files were selected. The types are determined by the extension of the geometry file. The names are determined by the filename. Any of the fields may be changed from the their default values. The `Remove` button will remove the currently selected model from the library. The `Done` button will close the model library creation GUI, load the new model library, and return to the main LADAR Simulator GUI.

A.3.4 Model Editor GUI

Model geometry files can be created in many different ways. The units used to define the models and the orientation in which a model rests can vary greatly from file to file.

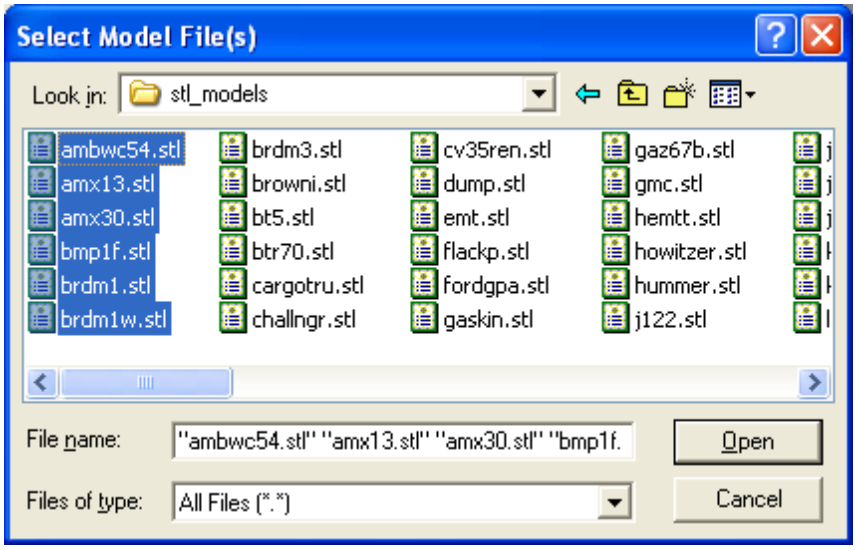


Figure 49: Dialog box for selecting CAD model files to add to the model library.

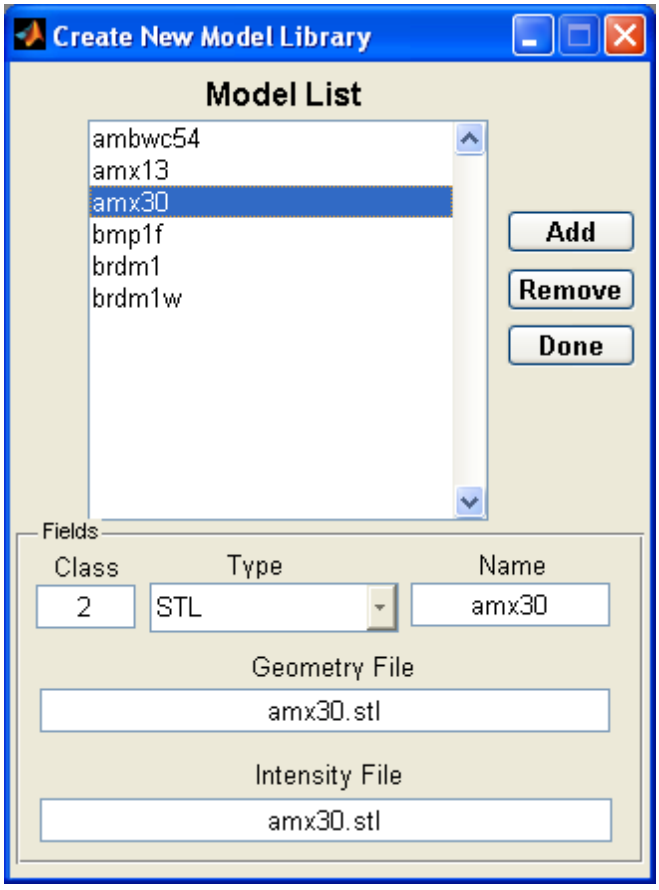


Figure 50: Model library creation GUI after adding CAD model files.

In addition to organizing a collection of model files, model libraries hold information about resizing, rescaling, and reorienting the models themselves. These adjustments allow you to redefine the default units and orientation of a model without permanently altering the geometry file. The Model Editor GUI can be launched by pressing the `Model Editor` button in the main LADAR Simulator GUI (see Figure 51).

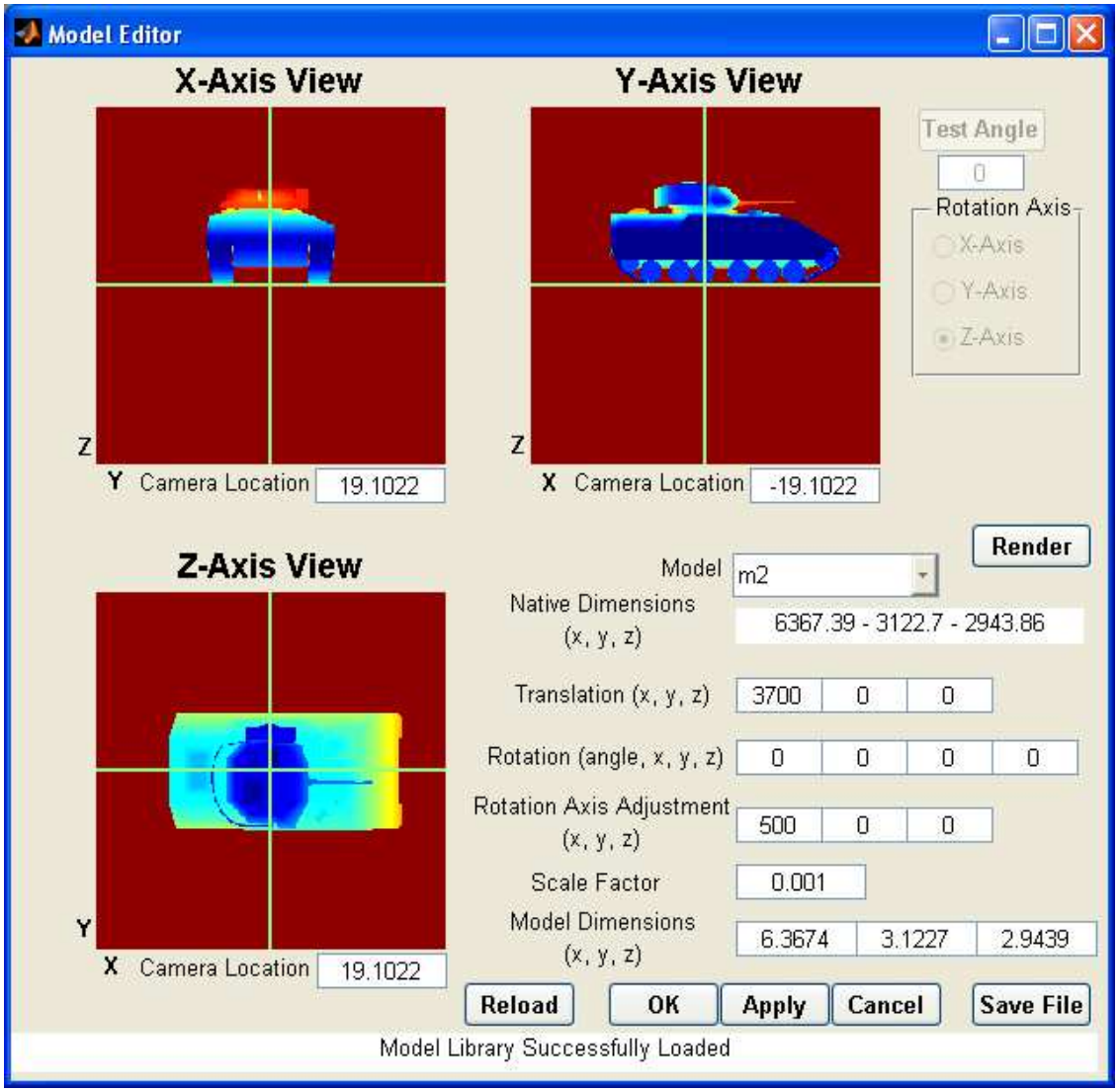
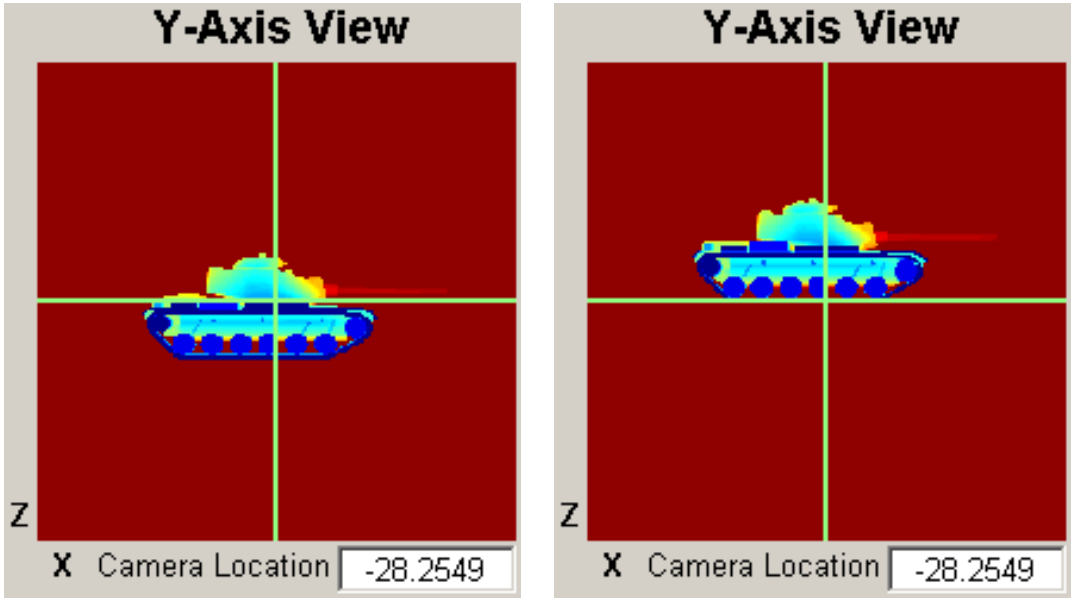


Figure 51: GUI for changing model library parameters.

The GUI has three windows, where each window faces the origin from one of the three coordinate axes. The GUI attempts to automatically adjust the viewing distance in each window so that the model is completely visible. If a different view

is desired, the value can be changed underneath the window. This value represents the distance between the camera and the origin along the specified axis. To "zoom in" on the object in the window, the magnitude of this value can be decreased. To "zoom out," the magnitude of this value can be increased.

The `Model` pop-up menu allows the user to choose the model to be adjusted. The `Translation` text fields are used to define a default translation to be applied before the scene-level rotations and translations. Sometimes, model files are defined in such a way that the origin is considered to be one of the bottom corners of the object. It may be convenient to place the origin on the bottom center of the model so that if a target is placed at $[0, 0, 0]$, then the target will straddle the center point. Using the *same units as the geometry file*, desired translations along the x, y, or z axes can be specified. An example is shown in Figure 52.



(a) (b)

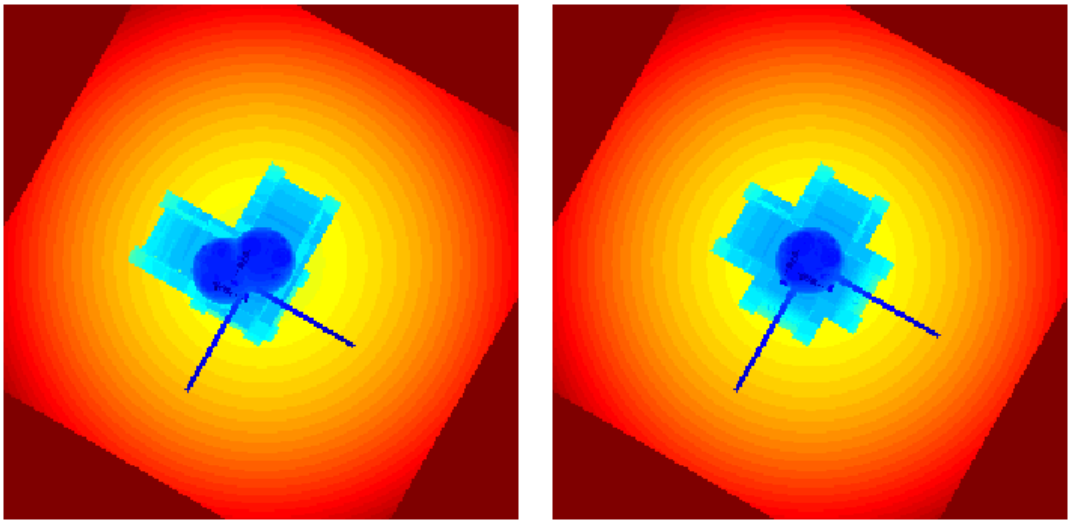
Figure 52: Images showing a tank before (a) and after (b) a translation adjustment along the z-axis. After the adjustment, tanks placed in a scene will not be halfway in the ground plane.

The `Rotation` text fields are used to define a default rotation to occur before scene-level rotations and translations. Sometimes model files are defined in a way

that leaves them in an unnatural orientation when used in a scene. For example, a car model may be sitting on its bumper by default. These text fields allow the user to define a rotation along any of the axes that will occur before a model is placed. In the car example, the user can define a rotation that places the car on its wheels. The rotations are specified in the angle-axis notation, where you specify a rotation angle and the vector form of an axis about which to rotate.

The `Rotation Axis Adjustment` text fields allow the adjustment of the position of a model before a rotation occurs. By default, rotations occur on an axis in the center of the model's bounding box. For symmetric models, this works out well. When there are asymmetric features, such as an extra long gun extending from a tank, it may be desirable to adjust the axis about which the rotation occurs. In the case of the long gun, the center of the bounding box may be closer to the front of the tank, and a rotation would appear awkward. An example of this effect is shown in Figure 53.

The `Native Dimensions` field shows the extent of the current model along the $x - y - z$ axes; these are the dimensions of the bounding box along each of these axes. The values in this field are determined by the native geometries of the model file, meaning that they are the same as defined in the CAD model geometry file. The `Model Dimensions` text field shows how the extents change after applying the value in the `Scale Factor` field. The scaling can also be adjusted by entering a new value directly in the x, y, or z `Model Dimensions` field. For example, a model's native dimensions are defined in such a way that if a vehicle is 5000 units long in the x-direction, and the vehicle is supposed to be four meters long in that direction, then the user can type "4" into the x field. All other `Model Dimensions` will adjust accordingly, as will the scale factor. The camera positions in the three windows will also be adjusted to provide the same view of the object after they have been re-sized. When adjusting the `Translation` or `Rotation Axis Adjustment` fields, the units



(a) (b)

Figure 53: Images showing how adjusting the rotation axis affects the final rotation. Both images show two T62 tanks, one at the default orientation and another at the same location but rotated by 90 deg. There is no rotation axis adjustment in (a) so the rotation axis is further from the center of the tank body. This makes the rotated tank appear as if it was displaced slightly. In image (b), the rotation axis is moved so that it is in the center of the tank body. This results in a more intuitive rotation.

are in the native model units, and not the scaled ones.

When using the Model Editor GUI, some trial and error is usually involved in order to obtain desired model adjustments.

A.3.5 Ground Plane GUI

The Ground Plane GUI provides users access to the geometric description of the ground plane (see Figure 54). The ground plane is a flat, rectangular surface. A corner is defined by setting the $[x, y, z]$ coordinates of the "Origin" field. The lengths in the x and y directions are specified using the next two text fields. If negative values are supplied in these fields, then the ground plane extends along the axes in the negative direction. An arbitrary intensity value is specified in the last text field. The value itself is arbitrary and may be set to any positive integer.

When changing fields, the GUI will perform a general validity check for the most recently used parameter and revert to the original value if the current value is found

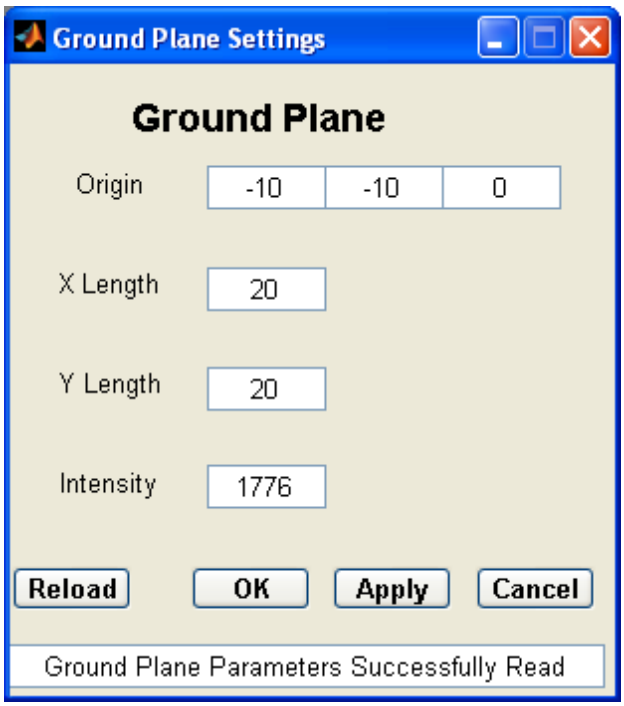


Figure 54: GUI for changing ground plane settings.

to be invalid. Pressing the `Reload` button will reset the GUI to its original state after it was first launched. Pressing the `OK` button will save the ground plane settings, close the Ground Plane GUI, and redraw the scene using the new ground plane. Pressing the `Apply` button will save the current ground plane settings and redraw the scene. Pressing the `Cancel` button will close the Ground Plane GUI and discard all changes since the last time the `Apply` button was pressed. Pressing the `Enter/Return` key after editing a text field simulates pressing the `Apply` button.

A.4 Library Interface

The functions used to read in models, create/manipulate scene configurations, and draw/display scenes can be called directly from a user-defined MATLAB script or function. This capability was included to allow users to run their own simulations using synthetic data generated on-the-fly.

Some of the functions and their descriptions are described here:

- Loading and saving files:
  - `load_configuration` - Load configuration file into structures
  - `load_data` - Retrieve image or point cloud from data file
  - `load_model_library` - Read model library file into a structure
  - `save_configuration` - Save configuration to a text file
  - `save_data` - Save a range image or point cloud to a data file
  - `save_model_library` - Save model library structure to a text file
- Manipulating configuration structures:
  - `add_target` - Add target with given parameters to configuration
  - `new_config` - Create an empty configuration structure
  - `remove_target` - Remove a target from a configuration
  - `set_ground_param` - Update the ground plane in a configuration
  - `set_target_param` - Update a single target in a configuration
- Manipulating view structure:
  - `new_view_settings` - Create a view settings structure
  - `set_view_param` - Update a view setting parameter
- Working with scene data:
  - `add_noise` - Adds noise to range imagery or point cloud data
  - `display_data` - Display a scene in a given figure or axes
- Scene creation:
  - `render_scene` - Create a range image or point cloud data set

Customized simulations typically begin with loading a model library and a configuration file (using `load_model_library` and `load_configuration`). If no configuration file is available, then a configuration is created from scratch with the `new_config` function with some ground plane information. Targets are added with the `add_target` function. The view is created and modified with the `new_view_settings` and `set_view_param` functions respectively, and so on.

The `ladar_simulator` folder contains a subfolder called `scripts` with an example simulation file named `make_multiview_ptc.m`. It creates a single point cloud from multiple views of the same scene, consisting of an object positioned in the middle of a ground plane. The sensor takes frames at 120 deg increments and merges the point clouds from the individual frames into a single point cloud. It then saves the point cloud and configuration structure to files using the `save_data` and `save_configuration` functions, respectively. Each range image is displayed along the way, as was well as the final point cloud, using the `display_data` function. This script may be used as a guide for creating other customized simulations.

A.5 File and Structure Formats

Although the GUIs and library interfaces should be sufficient for most users, some users may wish to directly manipulate the configuration files, model libraries, or the various structures. This section includes descriptions of these formats.

When manipulating the structures directly, it is important to remember data types. By default, MATLAB stores numbers as double precision floating point values. Since these structures are passed to C functions that utilize external libraries, many of the fields in these structures are not double precision floating point. Therefore, it is important to use the appropriate data type when assigning values to the structure fields referenced in this section.

A.5.1 Configurations

The configuration file and structure format includes fields for organizing different components of a scene. These components include target parameters, view settings, and the ground plane description. All arrays are stored row-wise.

A.5.1.1 Configuration Structure

The configuration structure description is shown in Table 2. It holds information on the targets and the ground plane in the scene.

Table 2: Configuration structure fields.

Field Name	Data Type	Description
numTargets	uint32 scalar	Number of targets in the scene
targets	structure array	Array of target structures
useGround	int32 scalar	-1 = Do not include a ground plane 0 = Use default ground plane 1 = Use customized ground plane defined in the configuration structure
groundIntensity	uint32 scalar	ground intensity (any nonnegative integer)
groundOrigin	double array	Three-element $[x, y, z]$ array representing the origin of the ground plane
groundXLength	double scalar	Length of the ground plane in the positive-x direction
groundYLength	double scalar	Length of the ground plane in the positive-y direction

A.5.1.2 Target Structure

The target structure description is shown in Table 3. The *class* field holds a nonnegative integer that identifies a unique model in the corresponding model library. The *position* field represents the target's $[x, y, z]$ coordinate position. The *orientation* is stored in angle-axis notation, so that rotations occur by rotating the target θ degrees along the $[x, y, z]$ axis. In most cases, rotations will occur about the z-axis, so the $[x, y, z]$ portion of this field should be set to $[0, 0, 1]$. Future revisions could support more intuitive rotations, like Euler angles. The *scale* field scales the dimensions of the

current target by a total value of s. It then scales the corresponding dimensions (as defined in the CAD model) by an additional x, y, or z in each of those directions. By default, this array should be set to $[1, 1, 1]$ for no additional scaling. The *visible* field is either set to zero or one and controls the visibility of the target in the configuration. The library function that manipulates the target structure is `set_target_param`.

Table 3: Target structure fields.

Field Name	Data Type	Description
class	uint32 scalar	Unique target class ID
position	single array	Three-element array of target $[x, y, z]$ coordinates
orientation	single array	Four-element array of target $[\theta, x, y, z]$ orientation
scale	single array	Four-element array of target $[s, x, y, z]$ scale
visible	uint32 scalar	0 or 1 specifying whether to render a target

A.5.1.3 View Settings Structure

The view settings are stored in a separate structure with parameters defined in Table 4. The library functions that manipulate the view structure are `new_view_settings` and `set_view_param`.

Table 4: View settings structure fields.

Field Name	Data Type	Description
vfov	double scalar	Vertical field of view
hfov	double scalar	Horizontal field of view
position	double array	Three-element array of sensor $[x, y, z]$ coordinates
lookAt	double array	Three-element array of "look at" $[x, y, z]$ coordinates
upv	double scalar	Three-element array representing the up direction
clipping	double array	Two-element array consisting of the minimum and maximum ranges
dataDim	uint32 array	Row x Column dimension of the result image in pixels, (the vertical and horizontal sampling)

A.5.1.4 Configuration File Format

Figure 55 shows the structure of a configuration file. Configuration files can be loaded and saved with the `load_configuration` and `save_configuration` functions,

respectively.

```
model_library <filename>

v_field_of_view <degrees>
h_field_of_view <degrees>
camera_position <x position> <y position> <z position>
object_position <x position> <y position> <z position>
up_vector <x position> <y position> <z position>
clip_distance <min range> <max range>
data_dims <vertical pixels> <horizontal pixels>

number_of_targets <nonnegative integer>

target 1
class_of_target <nonnegative integer ID>
position <x position> <y position> <z position>
orientation <degrees> <x axis> <y axis> <z axis>
scale <global value> <x position> <y position> <z position>

...

target <final target number>
class_of_target <nonnegative integer ID>
position <x position> <y position> <z position>
orientation <degrees> <x axis> <y axis> <z axis>
scale <global value> <x position> <y position> <z position>

use_ground <-1, 0 or 1>
intensity <nonnegative integer>
origin <x position> <y position> <z position>
x_length <value>
y_length <value>
```

Figure 55: Configuration file format.

A.5.2 Model Libraries

Model libraries are collections of model geometries and associated rendering parameters.

A.5.2.1 Model Library Structure

The Model Editor and New Model Library GUIs are currently the only means by which model libraries may be changed.

Model library structures contain two fields: `numModels` and `models`. `numModels` is a field of type uint32 that stores the number of models in the library structure. `models` is an array of model structures, defined in Section A.5.2.2.

A.5.2.2 Model Structure

The fields of a model structure are defined in Table 5. Section A.5.2.3 goes into more detail about the purpose of some of these fields. `gFile` and `rFile` will most likely be the same unless the CAD model is defined in the PRISM format. For CAD models that have no intensity information, the relevant fields are populated with arbitrary values when the models are first read from the files.

A.5.2.3 Model Library File Format

The model library file format is designed to keep a record of CAD models common to a certain scene and the adjustable parameters for each of those models. To load or save model library files, the `load_model_library` or `save_model_library` functions are used. The first line in the file contains the term `number_of_models` followed by a single space and an integer representing the number of models in the file. Each model is listed on consecutive lines, and the model parameters on a single line are separated by a single space. The parameters are as follows:

1. A unique nonnegative integer model identifier (in the configuration structure/file, this integer is the same as the class).

2. Type of CAD model (`STL` for ASCII or binary stereolithography files, `3DS` for binary 3D Studio files, `PRISM` for files in the Prism file format from Thermoanalytics, and `OFF` for Princeton shape benchmark files).

Table 5: Model structure fields and descriptions.

Field Name	Data Type	Description
class	uint32 scalar	numerical identifier for the model
type	char array	text description of the model type
name	char array	text identifier for the model
translate	single array	three-element (x, y, z) translation from the model's origin
rotate	single array	four-element (θ, x, y, z) default model rotation
rotateAdjust	single array	three-element (x, y, z) translation of the model rotation axis
scale	single scalar	global scaling of model units
gFile	char array	file and path of geometry file
rFile	char array	file and path of radiance file (if available), or `gFile`
vertexTable	single matrix	$N \times 3$ matrix of model vertices
vertexTableLength	uint32	N as used in `vertexTable`
facetIndexLists	cell array	$M \times 1$ array of uint32 arrays of indices into `vertexTable`
facetIndexListSizes	uint32 array	$M \times 1$ array containing the sizes of each array in `facetIndexLists`
numberOfFacets	uint32 scalar	the M as referenced above (number of facets in model)
intensityRegion	uint32 array	the intensity index of each facet
maxVertex	single array	three-element (x, y, z) for the largest bounding box coordinate
minVertex	single array	three-element (x, y, z) for the smallest bounding box coordinate
intensityList	uint32 array	$K \times 1$ array of intensity (or arbitrary) values
intensityListLength	uint32 scalar	K as referenced above (number of intensities)

3. Relative path (starting from the location of the model library file) and filename of the CAD model.

4. A repeat the previous path and filename for non-Prism files. For Prism files, this field contains the corresponding radiance file.

5. The number `0` (not used, but left in the code for legacy purposes).

6. A unique text string ID for the model. This will be displayed in the pop-up menus of the GUIs that allow that users to switch model types when creating a scene.

7. An array of three numbers in the form `[x,y,z]` (including the brackets and commas). This represents a coordinate translation from the model's origin as defined in the units that the model was created in. By default, this field is set to `[0,0,0]`. When changed, this field allows users to define another point in the model space as the origin. As an example, imagine rendering a model at the point $[0, 0, 0]$ in the scene coordinates. In many instances, this has the effect of rendering a lower corner of the model at that point. If the user would like for all models to be centered at the point of placement, the user could adjust the x and y parameters of this vector to first translate the model by these coordinates.

8. An array of four numbers in the form `[`θ`,x,y,z]` (including the brackets and commas). Similar to the previous field, this represents a standard initial rotation to take place before rendering a model. By default, this is set to `[0,0,1,0]`.

9. An array of three numbers in the form `[x,y,z]`, including the brackets and commas. This represents a translation to take place before the rotation occurs. By default, this may be set to `[0,0,0]`.

10. A scalar value that represents a scaling of the coordinates in the model file. By default, this may be set to `1`.

For an example model library file, see Figure 56.

```
number_of_models 3
0 OFF dir/mod1.off dir/mod1.off 0 mod1 [0,0,0] [0,0,0,0] [0,0,0] 1
1 OFF dir/mod2.off dir/mod2.off 0 mod2 [0,0,0] [0,0,0,0] [0,0,0] 1
2 OFF dir/mod3.off dir/mod3.off 0 mod3 [0,0,0] [0,0,0,0] [0,0,0] 1
```

Figure 56: Example of a model library file.

A.5.3 Data Sets

The LADAR Simulator can create data sets in two forms: range images and point clouds. Data sets can be saved or loaded with the functions `save_data` and `load_data`. Data is displayed in MATLAB with the function `display_data`.

A.5.3.1 Range Imagery

Range images are rectangular grids where each element represents the range from the sensor to world coordinate captured by that element. In MATLAB, these images are stored in matrix form. See Figure 57 for a typical range image.

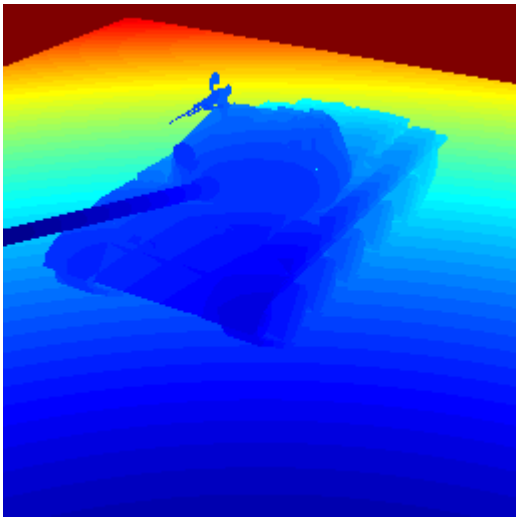


Figure 57: A typical range image.

A.5.3.2 Point Clouds

Point clouds are created by taking the two-dimensional points in a range image and converting them to their corresponding three-dimensional coordinates. The resulting point cloud, consisting of L points, is constructed as a $3 \times L$ matrix. If an image is rendered with a $N \times M$ data dimension, then there will be at most NM coordinates in the resulting point cloud. The number of coordinates L is less than NM when there is sky present in the scene, when a ground plane is not rendered, or when pixels are set at the minimum or maximum range values. A typical point cloud is shown in Figure 58. If the user wishes to create a single point cloud from multiple views, this can be easily accomplished by appending the new point clouds to the previous ones. For example, a 3×15000 point cloud and a 3×20000 point cloud from two different views of the same scene can be combined into a single 3×35000 point cloud.

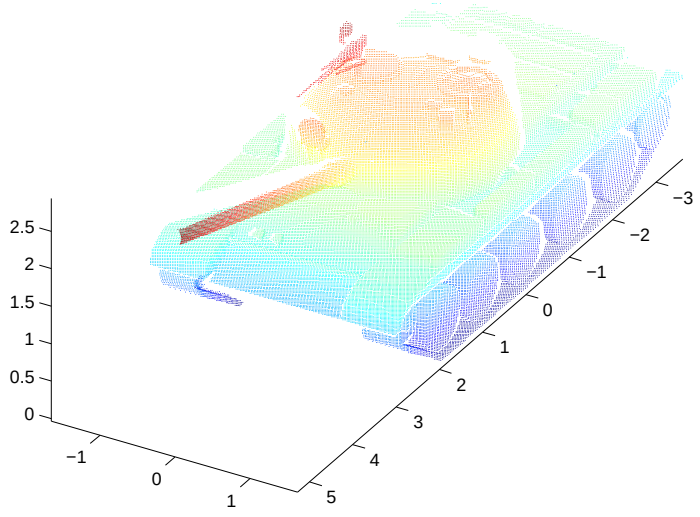
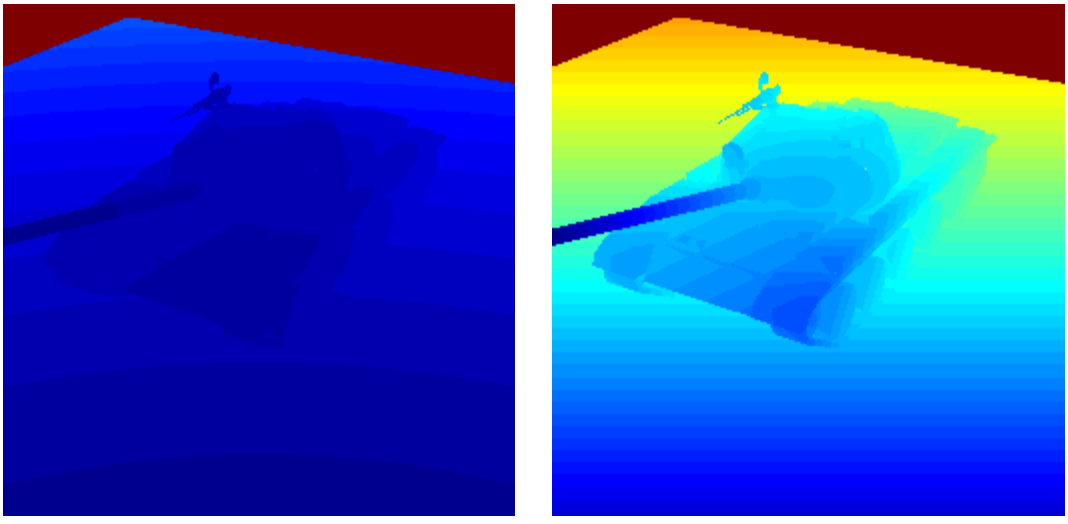


Figure 58: A typical point cloud.

A.5.3.3 Depth Buffer

If desired, the user can also create images that consist of the raw values pulled from the OpenGL depth buffering system after rendering the scene. This type of image is similar to the range image in that each pixel represents some notion of range, but the values will be in abstract double precision floating point position values along the camera view axis. This tends to be useful for visualizing data sets since depth is treated linearly without converting to true range value, so the default colormap will not obscure certain image features if the minimum and maximum ranges are set too far apart. Rendering is also faster since there is no conversion to actual range values, allowing 1) the GUIs to be more responsive to changes and 2) simulations to run faster. It is beneficial to work in this mode when setting up a scene. Figure 59 shows a side-by-side comparison of a range image and a depth buffer image with larger then necessary minimum and maximum range difference.



(a) (b)

Figure 59: Comparison of a range image and depth buffer image when minimum and maximum ranges are set at 0.1 and 100, respectively.

REFERENCES

[1] Angel, E., *OpenGL^{TM}: A Primer*. Boston, MA: Addison-Wesley, 2002.

[2] Ashlock, D., *Evolutionary computation for modeling and optimization*. Springer, 2006.

[3] Bounds, J. K., "The infrared airborne radar sensor suite," Report RLE Technical Report No. 610, Massachusetts Institute of Technology, December 1996.

[4] Braga-Neto, U., Choudhary, M., and Goutsias, J., "Automatic target detection and tracking in forward-looking infrared image sequences using morphological connected operators," *Journal of Electronic Imaging*, vol. 13, no. 4, pp. 802–813, 2004.

[5] Casasent, D. and Wang, Y.-C., "Automatic target recognition using new support vector machine," in *International Joint Conference on Neural Networks*, vol. 1, pp. 84–89, 2005.

[6] Chan, L. A., Der, S. Z., and Nasrabadi, N. M., "Dualband FLIR fusion for automatic target recognition," *Information Fusion*, vol. 4, no. 1, pp. 35–45, 2003.

[7] Clark, L. G., Perlovsky, L. I., Schoendorf, W. H., Plum, C. P., and Keller, T. L., "Evaluation of forward-looking infrared sensors for automatic target recognition using an information-theoretic approach," *Optical Engineering*, vol. 31, no. 12, pp. 2618–2627, 1992.

[8] Cooper, M. L., Grenander, U., Miller, M. I., and Srivastava, A., "Accommodating geometric and thermodynamic variability for forward-looking infrared sensors," in *Algorithms for Snythetic Aperture Radar IV* (Zelnio, E., ed.), vol. Proc. SPIE 3070, pp. 162–172, 1997.

[9] Cooper, M. L. and Miller, M. I., "Information measures for object recognition," in *Algorithms for Synthetic Aperture Radar Imagery V* (Zelnio, E., ed.), vol. Proc. SPIE 3370, pp. 637–645, 1998.

[10] Cover, T. M. and Thomas, J. A., *Elements of Information Theory*. Wiley Series in Telecommunications, New York, NY: John Wiley & Sons, Inc., 2nd ed., 1991.

[11] Dempster, A. P., Laird, M. N., and Rubin, D. B., "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

[12] Der, S., Chan, A., Nasrabadi, N., and Kwon, H., "Automated vehicle detection in forward-looking infrared imagery," *Applied Optics*, vol. 43, no. 2, pp. 333–48, 2004.

[13] Dixon, J. H. and Lanterman, A. D., "Toward practical pattern-theoretic ATR algorithms for infrared imagery," in *Automatic Target Recognition XVI* (Sadjadi, F. A., ed.), vol. Proc. SPIE 6234, 2006.

[14] Draper, B. A., Baek, K., Bartlett, M. S., and Beveridge, J. R., "Recognizing faces with PCA and ICA," *Computer Vision and Image Understanding*, vol. 91, no. 1-2, pp. 115–37, 2003.

[15] Duda, R. O., Hart, P. E., and Stork, D. G., *Pattern Classification*. New York, NY: John Wiley & Sons, Inc., 2nd ed., 2001.

[16] Gerwe, D. R., Hill, J. L., and Idell, P. S., "Cramér-Rao analysis of orientation estimation: Influence of target model uncertainties," *Journal of the Optical Society of America A: Optics and Image Science, and Vision*, vol. 20, no. 5, pp. 817–826, 2003.

[17] Gerwe, D. R. and Idell, P. S., "Cramér-Rao analysis of orientation estimation: Viewing geometry influences on the information conveyed by target features," *Journal of the Optical Society of America A: Optics and Image Science, and Vision*, vol. 20, no. 5, pp. 797–816, 2003.

[18] Green, P. J., "A primer on Markov chain Monte Carlo," *Complex stochastic systems (Eindhoven, 1999)*, vol. 87, pp. 1–62, 2001.

[19] Green Jr., T. J. and Shapiro, J. H., "Maximum-likelihood laser radar range profiling with the expectation-maximization algorithm," *Optical Engineering*, vol. 31, no. 11, pp. 2343–54, 1992.

[20] Green Jr., T. J. and Shapiro, J. H., "Detecting objects in three-dimensional laser radar range images," *Optical Engineering*, vol. 33, no. 3, pp. 865–74, 1994.

[21] Grenander, U., *Elements of Pattern Theory*. Baltimore, MD: Johns Hopkins University Press, 1996.

[22] Grenander, U., Miller, M. I., and Srivastava, A., "Hilbert-Schmidt lower bounds for estimators on matrix lie groups for ATR," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 790–802, 1998.

[23] Grenander, U., Srivastava, A., and Miller, M. I., "Asymptotic performance analysis of Bayesian target recognition," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1658–65, 2000.

[24] Grenander, U. and Miller, M., *Pattern Theory: From Representation to Inference*. New York, NY: Oxford University Press, 2007.

[25] Grenander, U. and Miller, M. I., "Representations of knowledge in complex systems," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 56, no. 4, pp. 549–603, 1994.

[26] Grünwald, P. D., *The minimum description length principle*. MIT press, 2007.

[27] Haeberli, P. and Akeley, K., "The accumulation buffer: hardware support for high-quality rendering," in *ACM SIGGRAPH Computer Graphics*, vol. 24, pp. 309–318, ACM, 1990. 4.

[28] Han, F., Tu, Z., and Zhu, S.-C., "Range image segmentation by an effective jump-diffusion method," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, no. 9, pp. 1138–1153, 2004.

[29] Hastings, W. K., "Monte carlo sampling methods using markov chains and their applications," *Biometrika*, vol. 57, no. 1, pp. 97–109, 1970.

[30] Higham, D. J., "An algorithmic introduction to numerical simulation of stochastic differential equations," *SIAM Review*, vol. 43, pp. 525–546, 2001.

[31] Holst, G. C., *Common Sense Approach to Thermal Imaging*. Winter Park, FL: JCD Publishing, 2000.

[32] Jacobs, P. A., *Thermal Infrared Characterization of Ground Targets and Backgrounds*, vol. TT70 of *Tutorial Texts in Optical Engineering*. Bellingham, Washington: SPIE Press, 2nd ed., 2006.

[33] Jain, A., Moulin, P., Miller, M. I., and Ramchandran, K., "Information-theoretic bounds on target recognition performance based on degraded image data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 9, pp. 1153–66, 2002.

[34] Johnson, K., Curran, A., Less, D., Levanen, D., and Marttila, E., "MuSES: A new heat and signature management design tool for virtual prototyping," in *Ninth Annual Ground Target Modeling and Validation Conference*, 1998.

[35] Kelley, C. T., *Iterative Methods for Optimization*, vol. 18 of *Frontiers in Applied Mathematics*. Philadelphia, PA: SIAM, 1999.

[36] Koksal, A. E., Shapiro, J. H., and Miller, M. I., "Performance analysis for ground-based target orientation estimation: FLIR/LADAR sensor fusion," in *Thirty-Third Asilomar Conference on Signals, Systems, and Computers*, vol. 2, pp. 1240–4, 1999.

[37] Koksal, A. E., Shapiro, J. H., and Wells, W. M., "Model-based object recognition using laser radar range imagery," in *Automatic Target Recognition IX* (Sadjadi, F. A., ed.), vol. Proc. SPIE 3718, pp. 256–266, 1999.

[38] Kwong, R. H. and Johnston, E. W., "A variable step size lms algorithm," *IEEE Transactions on Signal Processing*, vol. 40, no. 7, pp. 1633–42, 1992.

[39] Lanterman, A. D., "Jump-diffusion algorithm for multiple target recognition using laser radar range data," *Optical Engineering*, vol. 40, no. 8, pp. 1724–1728, 2001.

[40] Lanterman, A. D., Miller, M. I., and Snyder, D. L., "General Metropolis-Hastings jump diffusions for automatic target recognition in infrared scenes," *Optical Engineering*, vol. 36, no. 4, pp. 1123–37, 1997.

[41] Lanterman, A. D., "Bayesian inference of thermodynamic state incorporating Schwarz-Rissanen complexity for infrared target recognition," *Optical Engineering*, vol. 39, no. 5, pp. 1282–1292, 2000.

[42] Lanterman, A. D., "Schwarz, Wallace, and Rissanen: Intertwining themes in theories of model order estimation," *International Statistical Review*, vol. 69, no. 2, pp. 185–212, 2001.

[43] Levenberg, K., "A method for the solution of certain non-linear problems in least squares," *Quart. Appl. Math*, vol. 2, p. 164168, 1944.

[44] Li, B., Chellappa, R., Zheng, Q., Der, S., Nasrabadi, N., Chan, L., and Wang, L., "Experimental evaluation of FLIR ATR approaches - a comparative study," *Computer Vision and Image Understanding*, vol. 84, no. 1, pp. 5–24, 2001.

[45] Madsen, K., Nielsen, H. B., and Tingleff, O., "Methods for non-linear least squares problems," report, Technical University of Denmark, DTU, 2004.

[46] Marquardt, D. W., "An algorithm for least-squares estimation of nonlinear parameters," *SIAM Journal on Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[47] Miller, M. I., Grenander, U., O'Sullivan, J. A., and Snyder, D. L., "Automatic target recognition organized via jump-diffusion algorithms," *IEEE Transactions on Image Processing*, vol. 6, no. 1, pp. 157–74, 1997.

[48] Miller, M. I., Srivastava, A., and Grenander, U., "Conditional-mean estimation via jump-diffusion processes in multiple target tracking/recognition," *Signal Processing, IEEE Transactions on*, vol. 43, no. 11, pp. 2678–2690, 1995.

[49] Nikolic, M. M., Ortner, M., Nehorai, A., and Djordjevic, A. R., "An approach to estimating building layouts using radar and jump-diffusion algorithm," *Antennas and Propagation, IEEE Transactions on*, vol. 57, no. 3, pp. 768–776, 2009.

[50] Øksendal, B., *Stochastic differential equations*. Springer, 2003.

[51] Rissanen, J., *Information and complexity in statistical modeling*. Springer Science & Business Media, 2007.

[52] Rizvi, S. A. and Nasrabadi, N. M., "Fusion of FLIR automatic target recognition algorithms," *Information Fusion*, vol. 4, no. 4, pp. 247–58, 2003.

[53] Rynes, P., Curran, A., Johnson, K., Levanen, D., and Marttila, E., "New and improved signature modeling of ground vehicles with MuSES," report, ThermoAnalytics, Inc., 2000.

[54] Sadjadi, F. A., "Infrared target detection with probability density functions of wavelet transform subbands," *Applied Optics*, vol. 43, no. 2, pp. 315–323, 2004.

[55] Savage, J., Coker, C., Thai, B., Aboutalib, O., Chow, A., Yamaoka, N., and Kim, C., "Irma 5.2 multi-sensor signature prediction model," in *Modeling and Simulation for Military Operations II* (Schum, K. and Trevisani, D. A., eds.), vol. Proc. SPIE 6564, 2007.

[56] Segal, M. and Akeley, K., "The OpenGL graphics system: A specification (version 2.1)," Dec 2006.

[57] Shilane, P., Min, P., Kazhdan, M., and Funkhouser, T., "The princeton shape benchmark," in *Shape Modeling International*, pp. 167–178, IEEE, 2004.

[58] Srivastava, A., Grenander, U., Jensen, G. R., and Miller, M. I., "Jump-diffusion Markov processes on orthogonal groups for object pose estimation," *Journal of Statistical Planning and Inference*, vol. 103, no. 1-2, pp. 15–37, 2002.

[59] Storn, R. and Price, K., "Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[60] Sworder, D. D. and Boyd, J. E., "Jump-diffusion processes in tracking/recognition," *IEEE Transactions on Signal Processing*, vol. 46, no. 1, pp. 235–239, 1998.

[61] van Trees, H. L., *Detection, Estimation and Modulation Theory, Part I*. New York: John Wiley and Sons, 1968.

[62] Vasile, A. N. and Marino, R. M., "Pose-independent automatic target detection and recognition using 3D laser radar imagery," *Lincoln Laboratory Journal*, vol. 15, no. 1, pp. 61–78, 2005.

[63] Wallace, C. S., *Statistical and inductive inference by minimum message length*. Springer Science & Business Media, 2005.

[64] Zhu, S.-C., "Stochastic jump-diffusion process for computing medial axes in markov random fields," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 21, no. 11, pp. 1158–1169, 1999.

VITA

Jason Herbert Dixon was born and raised in Washington, D.C. He graduated from Georgetown Day High School in 1998. After graduating high school, Jason attended the University of Maryland Baltimore County (UMBC) as part of the Meyerhoff Scholarship Program. While at UMBC, Jason was a member of the National Society of Black Engineers and the Student Judicial Board. Jason graduated summa cum laude from UMBC in 2002, earning a Bachelor of Science degree in Computer Engineering. Before beginning his graduate studies, Jason participated in the National Institutes of Health (NIH) Postbaccalaureate Intramural Research Training Award (Postbac IRTA) program. While at NIH, Jason developed software tools for 3D reconstruction of molecules imaged by cryoelectron microscopy for the Laboratory of Structural Biology Research. Jason began graduate study at the Georgia Institute of Technology in 2003 as a recipient of the institute's President's Fellowship and a research assistantship from the School of Electrical and Computer Engineering. While at Georgia Tech, Jason was a member of the Black Graduate Students Association (BGSA), the Institute of Electrical and Electronic Engineers (IEEE), and SPIE, an international society for optics and photonics. In 2005, Jason earned a Master of Science degree in Electrical and Computer Engineering. Since 2008, Jason has been working full-time in Northern Virginia in the field of signal processing.